UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:

INGENIEROS ELECTRÓNICOS

TEMA:

DESARROLLO DE UN ENTRENADOR INALÁMBRICO DE PLANTAS DE CONTROL MEDIANTE UNA HERRAMIENTA DE PROGRAMACIÓN IOT

AUTORES:

DIANA CRISTINA YÁNEZ ZAMBRANO

PABLO ANDRÉS SUNTA ZAPATA

TUTOR:

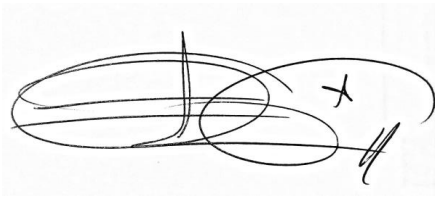
PILLAJO ANGOS CARLOS GERMÁN

Quito, Enero 2021

CESION DE DERECHOS DE AUTOR

Nosotros, Pablo Andrés Sunta Zapata y Diana Cristina Yánez Zambrano, con documentos de identificación N° 1725988206 y N° 1721712477 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: DESARROLLO DE UN ENTRENADOR INALÁMBRICO DE PLANTAS DE CONTROL MEDIANTE UNA HERRAMIENTA DE PROGRAMACIÓN IOT , mismo que ha sido desarrollado para optar por el título de: INGENIERO ELECTRÓNICO , en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.



…… ……

Pablo Andrés Sunta Zapata
C.I. No. 1725988206

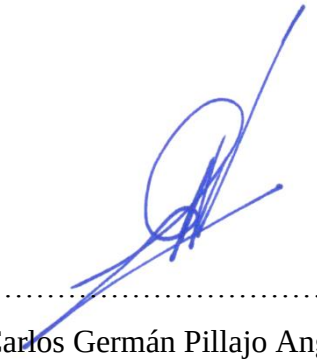
Diana Cristina Yánez Zambrano
C.I. No. 1721712477

Quito, Enero 2021

DECLARACIÓN DE COUTORIA DEL DOCENTE TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, DESARROLLO DE UN ENTRENADOR INALÁMBRICO DE PLANTAS DE CONTROL MEDIANTE UNA HERRAMIENTA DE PROGRAMACIÓN IOT, realizado por Sunta Zapata Pablo Andrés y Yánez Zambrano Diana Cristina, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, Enero 2021



……

Carlos Germán Pillajo Angos

C.I. No. 1709255119

DEDICATORIA

Este trabajo es dedicado a mis padres Guillermo Sunta y Mirian Zapata que, con su amor, apoyo incondicional, ejemplo, estudio y trabajo son el pilar fundamental de mi formación personal y académica, mis hermanos Diego y Anthony los cuales son mi compañía, inspiración e impulso para seguir adelante en mi vida.

Pablo

Dedico este trabajo de titulación a mis padres Carlos Yánez y Clara Zambrano quienes trabajan día y noche para que mi hermano y yo seamos unos buenos profesionales y unas buenas personas, y por qué ellos siempre me han apoyado incondicionalmente en todas mis metas, confiando en mí a cada momento.

También este trabajo se lo dedico a mi hermano Adam Yánez quien es mi otro impulso fundamental para ser una mejor persona cada día.

Diana

AGRADECIMIENTOS

Agradezco a mis tíos, primos, abuelos, por estar siempre presente en cada aspecto de mi vida ayudándome e impulsándome para ser un profesional y un buen ser humano, agradezco a mis amigos que, a pesar de las dificultades, hicieron que la estancia en la universidad sea muy agradable y que cada uno de ellos dejo una huella en mí.

Pablo

Quiero dar gracias a Dios por la oportunidad de culminar con éxito esta etapa de mi vida, de igual forma agradecer a toda mi familia y amigos por cada aliento para seguir adelante y un agradecimiento especial a mis padrinos y abuelita, porque jamás dejaron que me faltara algo en todos mis años de estudio y han sido incondicionales en todos los aspectos de mi vida.

Diana

Agradecemos a la Universidad Politécnica Salesiana, a los docentes y autoridades por abrir las puertas de sus prestigiosas aulas, ayudándonos a cumplir el sueño de culminar nuestros estudios universitarios, con bases en la ciencia, tecnología y valores humanos para triunfar en nuestras carreras profesionales, en la vida.

Además, agradecemos a nuestro tutor y amigo Ing. Carlos Pillajo quien nos motivó constantemente y ayudó con todos los medios para poder llevar a cabo este proyecto de titulación.

Los autores

ÍNDICE

1. CESION DE DERECHOS DE AUTOR ii

2. DEDICATORIA iv

3. AGRADECIMIENTOS v

4. ÍNDICE vi

5. RESUMEN xi

6. ABSTRACT xii

1. CAPÍTULO 1 ANTECEDENTES 1

1.1. PLANTEAMIENTO DEL PROBLEMA 1

1.2 JUSTIFICACIÓN 2

1.3 OBJETIVOS 3

1.3.1 Objetivo General 3

1.3.2 Objetivos Específicos 3

1.4 METODOLOGÍA 3

2. CAPÍTULO 2 FUNDAMENTACIÓN TEORICA 4

2.1 SISTEMAS DE CONTROL 4

2.1.1 Elementos de un sistema de control 5

2.1.2 Sistema de control de lazo cerrado 5

2.2 COMUNICACIÓN INALAMBRICA WI-FI 6

2.3 WIRELESS NETWORK CONTROL SYSTEM (WNCS) 6

2.4 INTERNET DE LAS COSAS (IOT) 7

2.4.1 Protocolo MQTT 7

2.4.2 Modelo de Publicación y Suscripción 8

2.5 EPC (ENTRENADOR DE PLANTAS DE CONTROL) 9

2.6 PLACA DE DESARROLLO NODE MCU ESP8266 9
2.7 NODE RED 10
2.8 TARJETA EMBEBIDA 11
3. CAPÍTULO 3 FUNDAMENTACIÓN TEORICA 12
3.1 DIAGRAMA DEL WEPC 12
3.1.1 Diseño electrónico de alimentación del WEPC 13
3.1.2 Planta de velocidad y posición 15
3.1.3 Planta temperatura y nivel 16
3.1.4 Planta de domótica 18
3.2 DISEÑO WEPC 19
3.3 DIAGRAMA DE CONTROL DEL WNCS 21
3.4 DIAGRAMA DE BLOQUES Y COMUNICACIONES EN EL WEPC 22
3.5 IMPLEMENTACIÓN COMUNICACIÓN MQTT ENTRE WEPC - RASPBERRY PI 3 23
3.5.1 Comunicación Raspberry PI 3 – MQTT 23
3.5.2 Comunicación NodeMCU ESP8266 – MQTT 24
3.5.3 Instalar Node-RED 26
3.6 PROGRAMACIÓN EN NODE-RED 27
3.6.1 Planta Motor - Servomotor 27
3.6.2 Planta de temperatura y nivel 29
3.6.3 Planta de domótica 32
4. CAPÍTULO 4 PRUEBAS Y RESULTADOS 36
4.1 FUNCIÓN DE TRANSFERENCIA PLANTA MOTOR DC 36
4.1.1 PID planta de Velocidad 38
4.2 FUNCIÓN DE TRANSFERENCIA PLANTA DE TEMPERATURA 41
4.2.1 PID planta de Temperatura 43
4.3 FUNCIÓN DE TRANSFERENCIA PLANTA NIVEL 45

5. CAPÍTULO 5 ANTECEDENTES 48

5.1 CONCLUSIONES 48

5.2 RECOMENDACIONES 49

6. REFERENCIAS 50

7. ANEXOS 52

INDICE DE FIGURAS

Figura 2. 1 Entrada y Salida de Proceso 4
Figura 2. 2 Sistema de control en lazo cerrado 6
Figura 2. 3 Un sistema de control en red inalámbrico 7
Figura 2. 4 Funcionamiento PUB/SUB 8
Figura 2. 5 Planta de Entrenamiento 9
Figura 2. 6 Placa de desarrollo NODE MCU 10
Figura 2. 7 Interfaz gráfica Node RED 10

Figura 3. 1 Esquema de conexiones WEPC 12
Figura 3. 2 Esquema didáctico del proyecto 13
Figura 3. 3 Circuito Electrónico de alimentación 14
Figura 3. 4 Circuito Electrónico switch (Inalámbrico - Externo) 14
Figura 3. 5 Circuito Electrónico de Señales de Sensores 15
Figura 3. 6 Esquemático Velocidad/Posición 16
Figura 3. 7 Planta de Velocidad y Posición 16
Figura 3. 8 Esquemático Temperatura/Nivel 17
Figura 3. 9 Planta de Temperatura 17
Figura 3. 10 Planta de Nivel 18
Figura 3. 11 Esquemático Domótica 19
Figura 3. 12 Planta de Domótica 19
Figura 3. 13 Distribución de elementos WEPC 20
Figura 3. 14 WEPC 20
Figura 3. 15 Sensores y Actuadores WEPC 21
Figura 3. 16 Diagrama de control WNCS 21
Figura 3. 17 Diagrama de bloques y comunicaciones WEPC 23
Figura 3. 18 Ventana de instalación pluggin 25
Figura 3. 19 Ventana de instalación tarjeta ESP8266 25
Figura 3. 20 Ventana de instalación librería 26
Figura 3. 21 Control PID en NODE-RED 28
Figura 3. 22 Programación Servomotor NODE-RED 28
Figura 3. 23 Motor DC y Servomotor 29
Figura 3. 24 Programación PID temperatura en NODE-RED 31
Figura 3. 25 Programación Nivel/Temperatura 31

Figura 3. 26 Programación Entradas domótica 33
Figura 3. 27 Programación Salidas domótica leds 34
Figura 3. 28 Programación Salidas domótica relé/buzzer 34
Figura 3. 29 Dashboard domótica 35

Figura 4. 1 Datos y Set Point Planta Motor DC 37
Figura 4. 2 Función de transferencia Planta de Motor DC 37
Figura 4. 3 PID Tuner Matlab para Planta de Velocidad (Motor DC) 38
Figura 4. 4 Valores Kp, Ki, Kd según PID Tuner 39
Figura 4. 5 Gráfica Resultante 39
Figura 4. 6 Kp, Ki, Kd finales 40
Figura 4. 7 Gráficas y PID resultante Motor PID 40
Figura 4. 8 Set point de 101 rpm ,62 rpm y 110 rpm 41
Figura 4. 10 Función de Transferencia Planta de Temperatura 43
Figura 4. 11 PID Tuner Matlab para Planta de Temperatura 44
Figura 4. 12 Valores Kp, Ki, Kd Temperatura 44
Figura 4. 13 Gráfica y Resultados PID de Temperatura 45
Figura 4. 14 Datos y Set Point Planta de Nivel 46
Figura 4. 15 Función de transferencia Planta de Nivel 47

INDICE DE TABLAS

Tabla 2. 1 Atributos de las tarjetas de desarrollo 11

Tabla 3. 1 Topics y Variables Motor-Servomotor 27
Tabla 3. 2 Topics y Variables Temperatura- Nivel 30
Tabla 3. 3 Topics y Variables Domótica 32

RESUMEN

Desarrollo de un entrenador inalámbrico de plantas de control (WEPC), es un banco de pruebas mediante el cual es posible obtener un modo de entrenar al usuario en los distintos tipos de control inalámbrico en las diferentes plantas, el cual consta de 5 plantas divididas en tres partes, cada una de estas es controlada por una tarjeta embebida NODEMCU ESP8266 con conexión Wifi, la primera tarjeta maneja el control de posición angular y velocidad de un motor, la segunda tarjeta tiene lo necesario para controlar las variables de nivel y temperatura y la tercera tarjeta está asociadas a señales de entrada-salida digitales y una entrada analógica, que sirven para controlar dispositivos domóticos. En cada una de las plantas está distribuido tanto sensores como actuadores conectados a su respectiva tarjeta embebida que brindarán la información del estado actual de la planta. La tarjeta embebida será la encargada de procesar las señales eléctricas de los sensores enviar los datos al bróker o servidor, recibir datos del mismo y emitir señales eléctricas a los actuadores. El servidor o bróker es una Raspberry Pi 3 un mini ordenador independiente con conexión Wifi y su respectivo software Node-Red, el cual recibe los datos, los procesa, toma decisiones según los algoritmos desarrollados por el usuario y envía estas decisiones a cada una de las plantas, también cuenta con una interfaz gráfica para observar y manipular cada una de las variables en sus diferentes plantas, la comunicación entre la planta y el servidor será desarrollada mediante protocolos de comunicación IOT.

ABSTRACT

Development of a wireless control plant trainer (WEPC), is a test bench through which it is possible to obtain a way to train the user in the different types of wireless control in the different plants, which consists of 5 plants divided into three parts, each one is controlled by a NODEMCU ESP8266 embedded card with WIFI connection, the first card handles the control of angular position and speed of a motor, the second card has what is necessary to control the variables of level and temperature and the third card is associated with digital input-output signals and an analog input, which serve to control home automation devices. Both sensors and actuators are distributed in each of the plants, connected to their respective embedded card that will provide information on the current state of the plant. The embedded card will be in charge of processing the electrical signals from the sensors, sending the data to the broker or server, receiving data from it and emitting electrical signals to the actuators. The server or broker is a Raspberry Pi 3 an independent minicomputer with WiFi connection and its respective Node-Red software, which receives the data, processes it, makes decisions according to the algorithms developed by the user and sends these decisions to each of the plants also have a graphical interface to observe and manipulate each one of the variables in their different plants. Communication between the plant and the server will be developed using IOT communication protocols.

INTRODUCCIÓN

El presente proyecto se realizó con fines académicos y de investigación, debido a que cada planta dentro del entrenador tiene la opción de ser manejada inalámbricamente o externamente para que el usuario, comprenda y analice los comportamientos que presentan los diferentes procesos y obtenga más información en el momento de su aprendizaje, a continuación, se describe el desarrollo del documento.

En el primer capítulo se expone el planteamiento del problema y los respectivos objetivos a desarrollar en el presente proyecto, según los requerimientos de un banco de pruebas para sistemas de control inalámbrico.

En el capítulo 2 se presenta de manera rápida los conceptos necesarios para la implementación del banco de pruebas tales como: lazos de control, comunicación inalámbrica con su respectivo protocolo e internet de las cosas, que permitan realizar aplicaciones donde se pueda manipular sistemas de control de velocidad, temperatura y domótica con su respectivo sistema de comunicación inalámbrica.

En el capítulo 3 se procede al diseño de la placa electrónica, implementación del protocolo de comunicación y el desarrollo de la interfaz gráfica de cada una de las plantas de control, con el fin de efectuar diferentes pruebas a su funcionamiento.

En el cuarto capítulo se desarrollan los ensayos en cada planta con sus respectivos sistemas de control con la función de transferencia asociada a cada sistema, para puntualizar los resultados adquiridos.

En el capítulo 5 se redacta las conclusiones y recomendaciones basado en las pruebas y los resultados que se aplicaron al proyecto, antes y después del diseño e implementación del sistema.

CAPÍTULO 1
ANTECEDENTES

1.1. PLANTEAMIENTO DEL PROBLEMA

El mundo actual está muy interesado en el desarrollo de aplicaciones, así como monitoreo, diagnóstico y control en los sectores medioambiental, médico, agrícola e industrial, con el objetivo de investigar para mejorar las condiciones ambientales y sociales de la comunidad en general, además, incrementar la calidad y productividad en los procesos industriales. (Archila Cordova, 2013, pág. 4)

Para el estudiante o docente representa un problema el no tener sistemas de automatización inalámbricos listos en hardware y software ya que requieren la conexión e interacción de numerosos sensores y actuadores de una forma segura, ligera y confiable para poder aprender o investigar en sistemas de control inalámbricos. El mercado actual solo ofrece entrenadores de control convencionales es decir controlados por señales eléctricas que no se adecuan a las nuevas tendencias de la tecnología mundial como es el IOT.

Según Rallo (2019), los procesos industriales requieren alta fiabilidad en su control, un elemento muy importante es la rapidez en la que se transmiten los datos para poder verificar en qué estado se encuentran las variables a manipular y emplear un determinado control para un correcto comportamiento, por su seguridad, bajo tiempo de retardo y fiabilidad las comunicaciones por cable son de uso tradicional en este tipo de proceso, pero las nuevas tecnologías dentro de los sistemas de automatización dirigen a investigar mecanismos que admitan realizar los mismos procesos, pero con la libertad y escalabilidad que brindan las comunicaciones inalámbricas como el Wireless.

Dentro de la ingeniería, los procesos industriales han usado el concepto de internet de las cosas, innovando desde la manera en que se comunican sensores con actuadores hasta la manera de procesar y almacenar cada dato obtenido en los diferentes procesos, causando con esto que los métodos académicos de aprendizaje actuales principalmente los que son sobre la teoría de control e instrumentación con IOT se vean muy limitados

en cuanto a desarrollo de proyectos y prácticas de laboratorio, por lo tanto un equipo electrónico que contenga los principales procesos con un control inalámbrico, optimizaría el tiempo en que se realizan dichos proyectos y se agilizaría el proceso de aprendizaje de los mismos.

Por lo anteriormente expuesto, se plantea la siguiente pregunta: ¿Existe un entrenador de sistemas de control inalámbrico que, mediante protocolos basados en programación en red, permitan ejecutar aplicaciones IOT, en donde se pueda efectuar el entrenamiento de algoritmos de control?

1.2 JUSTIFICACIÓN

En el 2009 National Instruments lanzo al mercado un entrenador de plantas de control (EPC), el cual "es un equipo electrónico que incluye sensores y actuadores típicos en los sistemas de instrumentación y control tales como temperatura, velocidad, posición". (National Instruments, 2016, pág. 5) El mismo que está diseñado para conectarse a un ordenador por medio de una tarjeta de adquisición de datos DAQ, con el objetivo de facilitar el aprendizaje de conceptos de teoría de control e instrumentación.

La necesidad de acondicionar un proceso académico más dinámico en el área sistemas de control con IOT, requiere de una conexión teórico-práctico que enfoque al usuario a ser una persona capacitada a la hora de enfrentar problemas reales, es por eso que el entrenador propuesto conecta al usuario a formar parte de la solución a problemas reales tangibles para aplicar la teoría de control, aprovechando este recurso a través de la elaboración de prácticas de laboratorio y proyectos. Bajo este objetivo, se encontró una gran utilidad en desarrollar una tarjeta electrónica que maneje las principales plantas que se encuentran en los procesos industriales, para aprender los diferentes tipos de control y bajo las nuevas tendencias tecnológicas, como la comunicación inalámbrica y el internet de las cosas. Por lo tanto, se hace imprescindible el desarrollo de un entrenador didáctico de plantas para sistemas de control en red inalámbricas que permitirá el estudio y la rápida implementación de proyectos de control, además el usuario podrá enlazar diferentes procesos al mismo tiempo y de esta manera se contribuye con el aprendizaje de la nueva teoría de WNCS.

1.3 OBJETIVOS

1.3.1 Objetivo General

Desarrollar un prototipo de entrenador inalámbrico de planta de control mediante una herramienta de programación IOT para el entrenamiento de algoritmos de control.

1.3.2 Objetivos Específicos

- Elaborar plantas de control de velocidad, temperatura, domótica con sus respectivos sensores y actuadores, utilizando 3 controladores en todo el entrenador para su respectivo sistema de control inalámbrico.
- Implementar un sistema de comunicación inalámbrico mediante el protocolo MQTT (Message Queing Telemetry Transport) para el control de las diferentes plantas.
- Gestionar los datos obtenidos de las plantas a través de un sistema embebido inalámbrico para su interpretación, ejecución y procesamiento.
- Realizar pruebas de control inalámbrico aplicando algoritmo de control PID para la comprobación de su correcto funcionamiento.

  +

1.4 METODOLOGÍA

Con la metodología investigativa se tomarán artículos referenciales para estudiar el comportamiento de sistemas que integren plantas velocidad, temperatura y control por redes inalámbricas, con la intención de determinar los recursos necesarios para su implementación.

Con la metodología cuantitativa se realizará la medición de las diferentes variables que presentan cada una de las plantas este dato será enviado hacia el controlador inalámbrico para realimentar el sistema.

Con la metodología exploratoria del tiempo de respuesta de los sistemas de control implementados.

CAPÍTULO 2
FUNDAMENTACIÓN TEORICA

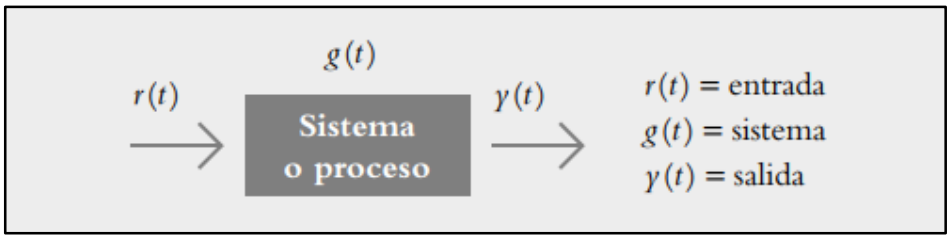
En el presente capítulo se estudia los fundamentos teóricos que se deben tomar en cuenta para el desarrollo correcto del entrenador, utilizando los principios de la Teoría de Control y el Internet de las Cosas. Como lo son los conceptos de Sistemas de control, sus elementos para el desarrollo apropiado del lazo de control del WEPC, también se investiga sobre la comunicación inalámbrica, los nuevos conceptos para trabajar con IOT y las placas de desarrollo con Wi-Fi, a su vez se realiza una comparación de tarjetas embebidas para explicar el uso de la Raspberry Pi3.

2.1 SISTEMAS DE CONTROL

"Un sistema de control automático es una interconexión de elementos que conforman una configuración señalada como sistema, de tal manera que el ajuste resultante es capaz de controlarse por sí mismo". (Gaviño, 2010)

Según Gaviño (2010) un sistema o un integrante del sistema a ser controlado, al cual para obtener una respuesta o salida y(t), se le aplica una señal r(t) en condición de entrada en todo caso puede representarse mediante bloques. (figura 2.1)

Figura 2. 1 Entrada y Salida de Proceso



Fuente (Gaviño, 2010)

El enlace que existe entre la entrada y salida es una correlación de causa y efecto con el sistema, en modo que el proceso a controlar nombrado "planta" vincula la salida con la entrada como lo indica Gaviño (2010). Son necesarios tres factores, para la conformación de los procesos automáticos actuales como lo indican Contreras, Tristancho y Vargas (2015).

- Sensores los mismos que receptan el estado del sistema
- Actuadores los cuales emiten señales de control
- Unidades de control que ejecutan el programa y toman decisiones

2.1.1 Elementos de un sistema de control

Morales y Ramírez (2013) señalan que aparte del sistema dinámico a ser controlado, un sistema de control tiene los siguientes elementos:

Empezando por la "planta" la cual es el sistema a controlar. Teniendo en cuenta que cada sistema se representa en base a su formulación matemática. Siendo así que una parte predomínate del sistema, sea la dinámica del mismo, permitiendo normas o leyes de control funcionales y adecuadas para un buen desempeño. Los actuadores también son componentes de un sistema de control y son dispositivos físicos que efectúan las acciones de control. Conocido de igual forma como aquel elemento que toma la energía del sistema y lo transforma en el movimiento, tendencia o esfuerzo físico.

De igual forma los sensores (lm35, encoder, sensores ultrasónicos, entre otros) que permiten al dispositivo electrónico, mediante post-procesamiento de sus lecturas, conocer su posición (al igual que otros valores) con respecto a un punto de referencia, el cual es de un origen fijo de coordenadas, que se pueden considerar como el origen de coordenadas del sistema, sabiendo que puede ser local o global.

Y por último el controlador es el elemento del sistema que ejecuta el cálculo de la salida de control necesaria para que así el sistema llegue al estado que se desea.

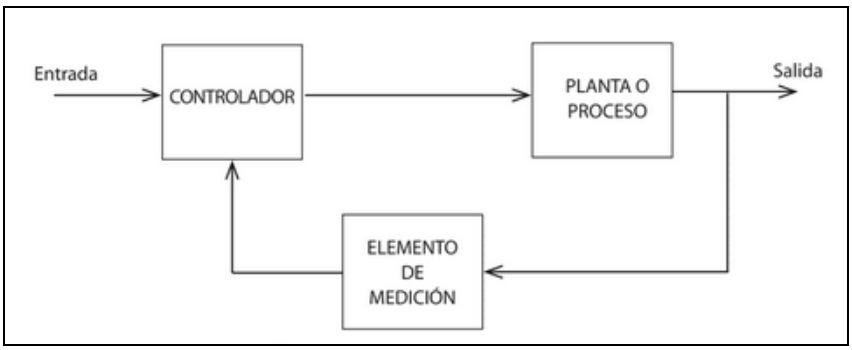
2.1.2 Sistema de control de lazo cerrado

Vázquez, Cardona y Leal (2015) definen que la señal de salida con acción directa sobre el control es el denominado Control de lazo cerrado, también llamados como control realimentado ya que esta señal es la encargada de efectuar el resultado de comparación o el error, en donde la salida del comparador, ingresa al control y se resuelve para reducir dicho error obteniendo así el set point o valor deseado en la salida del sistema. La expresión "lazo cerrado" va muy sujetada a la acción de realimentación por lo que procura reducir al mínimo el error. (p.21)

El controlador, el elemento de mando, sensores, actuadores y el proceso a ser

controlado son los elementos que abarca el sistema en lazo cerrado que se observa en la figura 2.2 que, según Vázquez, Cardona, & Leal (2015) son . (p.22)

Figura 2. 2 Sistema de control en lazo cerrado



Elementos de un sistema de control en lazo cerrado. Fuente: (Vázquez, Cardona, & Leal, 2015)

2.2 COMUNICACIÓN INALAMBRICA WI-FI

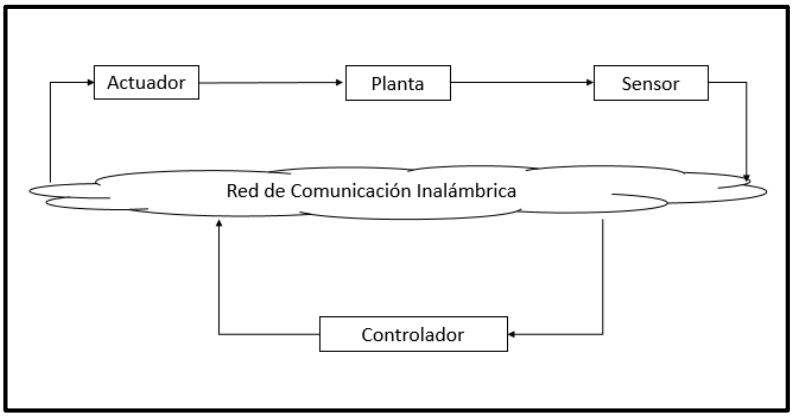
La comunicación inalámbrica como su nombre lo menciona, es aquella que para su interconexión no se utilizan cables, convirtiéndose en la tecnología de preferencia de todo tipo de usuarios. Las redes inalámbricas Wifi en empresas, industrias o para uso personal van en incremento según Carballar Falcón (2010).

"Wi-fi es una tecnología que accede que una gran variedad de equipos informáticos, puedan interconectarse sin la utilización de cables y la aplicación principal que está teniendo Wi-Fi en la actualidad es la de permitir que varios ordenadores de casa, oficina o industrias puedan compartir el acceso a internet. No obstante, este tipo de comunicación permite crear una red entre los distintos equipos para que se puedan compartir todos sus recursos." (Carballar Falcón, 2010).

2.3 WIRELESS NETWORK CONTROL SYSTEM (WNCS)

Dentro de los sistemas de control los datos e información emitidos por el sensor necesitan ser enviados hacia el controlador, al igual que los datos o información de este requieren ser enviados hacia el actuador. Sin embargo, en la actualidad no podemos ignorar la gran variedad de redes digitales disponibles que existen, motivo por el cual hay un creciente interés sobre los sistemas de control en red dijo Pillajo en 2018. WNCS es un sistema de control en lazo cerrado que considera todos los elementos en la red inalámbrica para la transmisión de datos.

Figura 2. 3 Un sistema de control en red inalámbrico



Elaborado por: Pablo Sunta y Diana Yánez

2.4 INTERNET DE LAS COSAS (IOT)

"El «Internet de las Cosas» (IoT) está dentro de las últimas innovaciones tecnológicas, y está fundamentada en la conexión de objetos habituales a la red, que procesan, intercambian y agregan datos sobre su entorno físico para suministrar y aumentar servicios a los usuarios finales. Además el IoT reconoce eventos o cambios, de tal manera los sistemas pueden comportar de forma apropiada y autónoma". (Barrio, 2018)

Su propósito es ofrecer una infraestructura que sobrease la restricción entre los objetos del mundo físico y su conceptualización en los métodos de información con el Internet, menciona así Barrio (2018).

2.4.1 Protocolo MQTT

MQTT es un protocolo de conectividad de M2M, tecnología que admite la comunicación de dos dispositivos, manejado ampliamente en IoT y está ganando popularidad en aplicaciones móviles y web. MQTT es un protocolo que funciona con un mecanismo de publicación-suscripción y se ejecuta sobre el protocolo TCP / IP. Es más ligero que el protocolo HTTP y, por lo tanto, es una opción muy interesante siempre que necesite enviar y recibir datos en tiempo real con un modelo de publicación-suscripción y necesite la huella más baja posible como indica Hillar (2017).

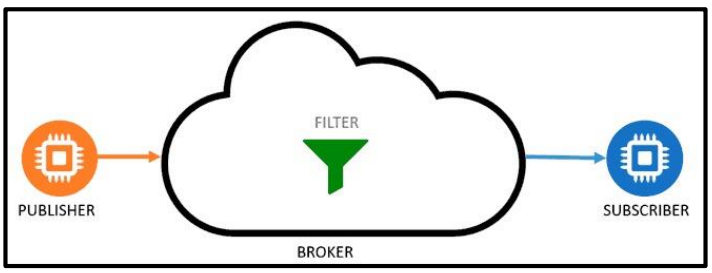
Según Gil (2018) las Ventajas de este protocolo son los siguientes:
Iniciando con la confiabilidad ya que trabaja sobre TCP, el protocolo que garantiza que los mensajes son entregados a su destino en la misma estructura que se emitieron y sin fallos. Siguiendo con la fiabilidad por que el protocolo MQTT permite la implementación de diferentes calidades de servicio que acceden a identificar pérdidas de mensajes o duplicidad en los mismos. De igual manera soporta el control de acceso debido a que sobrelleva el uso de credenciales para limitar el acceso a los requerimientos del bróker intermedio a aquellos clientes que estén autorizados. Y también hay una desventaja que hay que tomar en cuenta dijo Gil (2018) la cual hace referencia a que el protocolo es centralizado, donde un único equipo realiza todo el procesamiento de la información que se intercambia entre los diferentes participantes.

2.4.2 Modelo de Publicación y Suscripción

MQTT es el protocolo que determina dos tipos de elementos en la red: un bróker de mensajería y clientes. Al servidor se le denomina Bróker el mismo que recepta todos los mensajes de los clientes, como resultado, redirige estos mensajes a los clientes de la meta asignada. Un cliente es todo tipo de dispositivo que se pueda comunicar con el bróker y recibir mensajes. Un sensor de IoT en campo o una aplicación que procesa datos, puede ser un cliente. (Yuan, 2017)

- El cliente se enlaza al servidor o bróker. Tiene la capacidad de suscribirse a cualquier "topic" de mensajería del servidor. Este vínculo puede ser una comunicación con protocolo TCP/IP simple o una conexión TLS.
- El cliente publica los mensajes en un “topic”, enviando el contenido del mensaje y el “topic” al bróker.
- Finalmente, el servidor envía el mensaje a todos los clientes que se suscriben a este “topic”.

Figura 2. 4 Funcionamiento PUB/SUB



Fuente: (Llamas , 2019)

2.5 EPC (ENTRENADOR DE PLANTAS DE CONTROL)

El Entrenador de Planta de Control también denominado "EPC" es un módulo electrónico que incorpora varios actuadores y sensores comunes en los sistemas de control, dicho dispositivo fue también diseñado con el fin de reducir el tiempo de laboratorio de cátedras técnicas, el entrenador ofrece ensayos y pruebas de adquisición, procesamiento de señales y de control como describen en National Instruments (2016).

Figura 2. 5 Planta de Entrenamiento

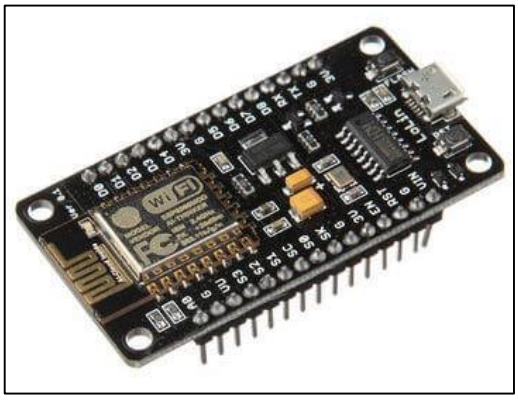
Fuente: (National Instruments, 2016)

2.6 PLACA DE DESARROLLO NODE MCU ESP8266

Según Del Valle Hernández (2017) el Node MCU es una placa de desarrollo que al igual que Arduino, es abierta a nivel de software y hardware lo que significa que facilita la programación en el microcontrolador llamado MCU (del inglés Microcontroller Unit), existen muchas versiones de estos dispositivos electrónicos y la versión más actual es la denominada ESP8266.

Dentro del Node MCU se encuentra el chip Wi-Fi ESP8266 económico que actúa por medio del protocolo TCP/IP. También incorpora un microcontrolador para operar el protocolo y el software preciso para soportar la conexión Wi-Fi. De la misma forma, la mayoría de modelos contienen entradas y salidas digitales de propósito general, tal como una entrada analógica de 10bit. Su característica más fuerte es poseer una conexión Wi-Fi en un microcontrolador con la capacidad de programar directamente con el entorno de Arduino con lo que es el ESP8266 es ideal para impulsar el desarrollo de aplicaciones de IoT. (Laborda, 2016)

Figura 2. 6 Placa de desarrollo NODE MCU



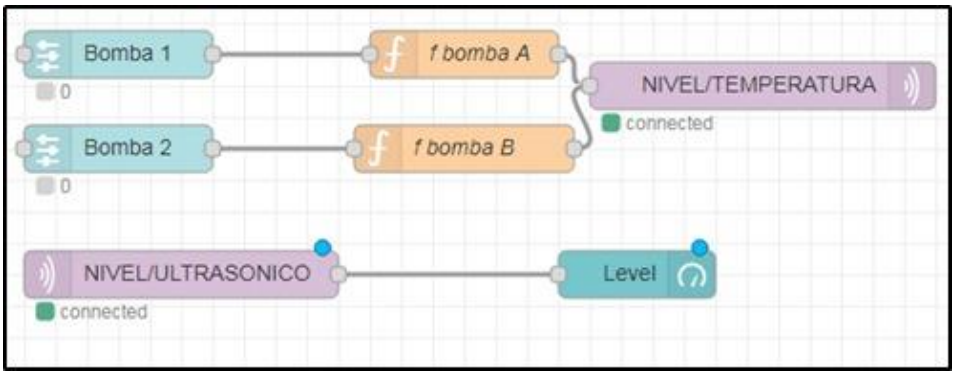
Fuente: (Del Valle Hernández, Programar Facil, 2017)

2.7 NODE RED

"Node-RED es una herramienta de software de programación muy potente que sirve para comunicar hardware y servicios de una forma muy rápida y sencilla. Simplifica la tarea de programar del lado del servidor gracias a la programación visual." (Del Valle Hernández, Programar Facil, 2019)

Nick O'Leary y Dave Conway-Jones del grupo de Tecnologías Emergentes de IBM en el año 2013 son los creadores de este software. Trabaja con nodos que son la configuración básica, dichos nodos se arrastran y manipulan a lo largo de la interfaz gráfica y nos concede realizar una tarea concreta. Recepta una llamada HTTP, un mensaje MQTT o un cambio de estado de un elemento señala Del Valle Hernández (2019) y estos nodos se ordenan en flujos denominados en inglés como "flows", que reúne nodos que se enlazan entre ellos. Todo este proceso esta basado en conexiones visuales, optimizando la programación.

Figura 2. 7 Interfaz gráfica Node RED



Fuente Pablo Sunta y Diana Yánez

2.8 TARJETA EMBEBIDA

Torrente (2016) define una tarjeta embebida como un circuito impreso que de acuerdo a sus características contiene un microcontrolador o un microprocesador, los responsables de realizar el procesamiento de datos, las diferentes operaciones que se ejecutan durante el procesamiento son controladas por un programa, el mismo que enseña al microprocesador lo que debe realizar, si se apela a un ejemplo como, mostrar información en una salida , leer el valor de un sensor, enviar bits de datos, de la misma forma una tarjeta embebida tiene una virtud muy importante con respecto a los microcontroladores ya goza de pines, que pueden ser entradas o salidas, al mismo tiempo cuenta con leds indicadores.

Examinando la tabla 2.1 se analizan los atributos de distintas tarjetas de desarrollo donde las diferencias más importantes son en el procesador y la memoria. De tal forma que se requiere de abundantes recursos para garantizar fluidez en el video, estas particularidades son las que conceden la adquisición y procesamiento de la imagen. (Gonzales, 2016)

Tabla 2. 1 Atributos de las tarjetas de desarrollo

ELEMENTO	**ARDUINO UNO**	**Raspberry PI 1-B**	**Raspberry PI 3-B**
Velocidad de reloj	16 MHZ	700 MHZ	1,2 GHZ
# núcleos	1	1	4
RAM	0,002 MB	512MB	100MB
CPU	No	ARM1176JZF-S	ARM Cortex-A53
SOC	ATmega328P	Broadcom BCM2835	Broadcom BCM 2837
Flash	32KB	Tarjeta SD (2-16 Gb)	Tarjeta microSD (2-32 Gb)
SO	Ninguno	Distribuciones Linux	Distribuciones Linux
Cámara	NO	No	5Mp
Precio	22,88	34,19	51,29
Tamaño	7,6X1,9X6,4cm	8,6X5,4X1,7cm	8,6X5,4X1,7cm
Peso	40 gr	45 gr	45gr

Detalle de las características de los sistemas embebidos para seleccionar el ideal para el sistema de visión artificial, Fuente: (Gonzales, 2016)

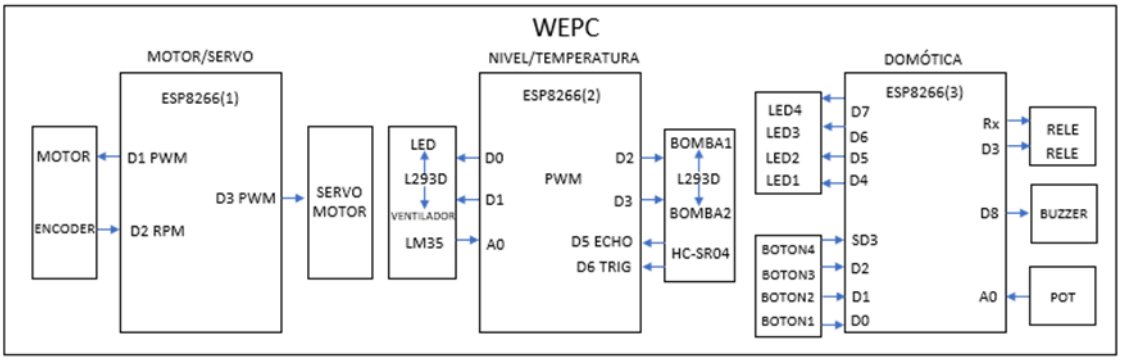
CAPÍTULO 3
FUNDAMENTACIÓN TEORICA

El capítulo detalla el diseño e implementación del entrenador inalámbrico de plantas de control mediante una herramienta de programación IoT. Para poder llevar a cabo el objetivo es importante e indispensable revisar de forma completa cada uno de los elementos que llevan las diferentes plantas para poder interconectar eficientemente el diseño eléctrico y de control del equipo.

3.1 DIAGRAMA DEL WEPC

El WEPC el cual tiene de 5 plantas divididas en tres partes, cada una de estas es controlada por una tarjeta embebida NODEMCU con conexión WiFi, la primera tarjeta maneja el control de posición angular y velocidad de un motor, la segunda tarjeta tiene lo necesario para controlar las variables de nivel y temperatura y la tercera tarjeta está asociadas a señales que sirven para controlar dispositivos que trabajan en domótica y en cada una de las plantas están distribuidos tanto sensores como actuadores.

Figura 3. 1 Esquema de conexiones WEPC

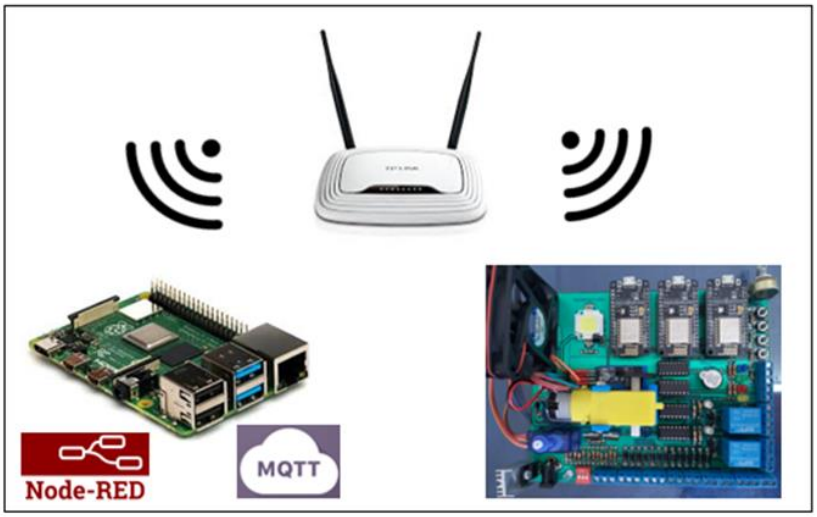


Esquema general de conexones y disribucion de elementos del WEPC . Elaborado por: Pablo Sunta y Diana Yánez

La tarjeta embebida es la encargada de procesar las señales eléctricas de los sensores, enviar los datos al bróker o servidor, recibe datos del mismo y emite las señales eléctricas a los actuadores. El servidor o bróker es una Raspberry Pi 3 que es un mini ordenador independiente con conexión WiFi y su respectivo software Node-Red este recibe los datos, los procesa, toma decisiones según los algoritmos desarrollados por el usuario y envía estas resoluciones a cada una de las plantas.

Cuenta con una interfaz gráfica la cual permite observar y manipular las diferentes variables, la comunicación entre cada planta y el servidor se basa en el protocolo MQTT del Internet de las Cosas.

Figura 3. 2 Esquema didáctico del proyecto



Elaborado por: Pablo Sunta y Diana Yánez

Como características principales del entrenador tenemos que se alimenta con 12[V] y 2000[mA], sus medidas físicas son 13 cm de alto y 21 cm de ancho.

El WEPC tiene dos modos de operación inalámbricamente mediante WIFI y la Raspberry pi, externamente mediante las borneras con un controlador independiente al módulo. El modo de operación se selecciona con tres interruptores colocados en la parte inferior izquierda uno para cada NodeMCU ESP8266.

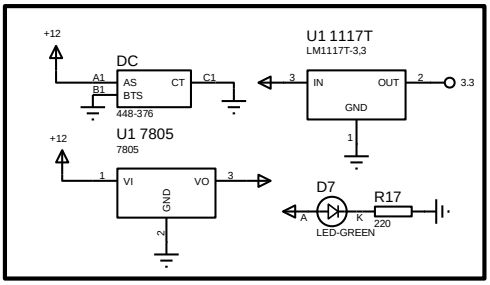
3.1.1 Diseño electrónico de alimentación del WEPC

Para la alimentación del WEPC tiene un DC Power Jack hembra, el entrenador trabaja con tres valores de voltajes:

- 12 [V] alimentación general del WEP, ventilador, motor DC, led 10 [W]
- 5 [V] integrados electrónicos, servomotor SG-90, bombas de agua, buzzer, leds, botones, relés, encoder,
- 3.3 [V] NodeMCU ESP8266, sensores conectados al NodeMCU

Se utiliza 2 reguladores de voltaje LM7805 y 1117T para para regular a 5[V] y 3.3 [V] respectivamente, además cuenta con un led para indicar cuando el WEPC esté conectado a la corriente

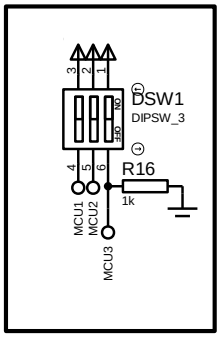
Figura 3. 3 Circuito Electrónico de alimentación



Elaborado por: Pablo Sunta y Diana Yánez

Para escoger el modo de operación inalámbrico y externo se utilizó un dip switch de 3 posiciones y así activar cada uno de los NodeMCU ESP8266 según su requerimiento.

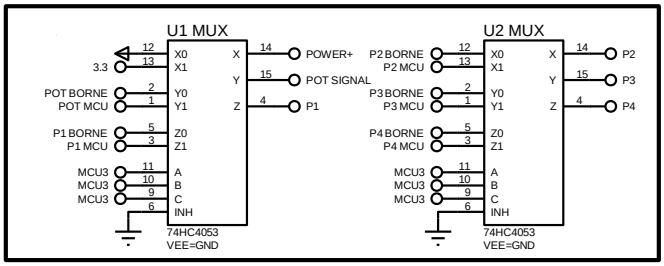
Figura 3. 4 Circuito Electrónico switch (Inalámbrico - Externo)



Elaborado por: Pablo Sunta y Diana Yánez

Como se mencionó los modos de operación del WEPC, se tiene un trato especial en las señales de los sensores exclusivamente en la planta de domótica, cuando está en modo externo los sensores es decir botones y potenciómetro trabajan con un voltaje de 5[V] con conexión a las borneras, pero como sabemos el Node MCU ESP8266 tiene un voltaje de operación de 3.3 [V] y todos sus periféricos deben trabajar con el mismo voltaje entonces se utiliza un multiplexor analógico 74HC4053 el cual permite escoger en que voltaje trabajen los señores o 5 [V] o 3.3 [V] y llegue el voltaje adecuado a los pines del NodeMCU, el encargado de dar la señal en que voltaje trabajar es el dip switch.

Figura 3. 5 Circuito Electrónico de Señales de Sensores



Elaborado por: Pablo Sunta y Diana Yánez

3.1.2 Planta de velocidad y posición

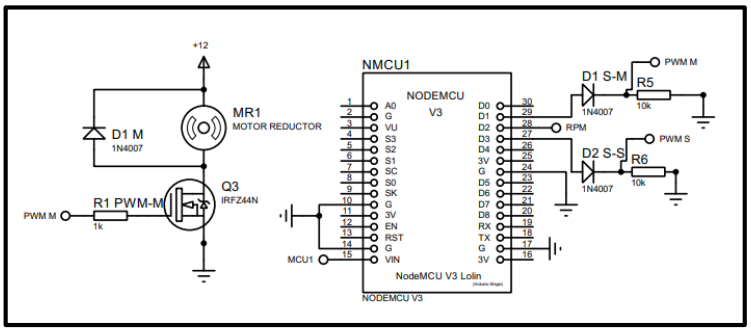
En esta planta el controlador consta deun motor reductor y un servomotor, que están a cargo de manipular las variables de velocidad y posición respectivamente, dichas variables se controlan desde la interfaz gráfica que es el Dashboard del software NodeRED, siendo el usuario el que escribe cuantas son las revoluciones por minuto que desea para el control de velocidad del motor , permitiendo analizar la curva de RPM la cual se ve dibujada en dicha interfaz y se actualiza cada que hay un cambio. Para el control de posición angular en el Dashboard existe una slider, que habilita la opción de controlar el ángulo al que el usuario quiera, teniendo un rango de 0° a 180° grados. Su diseño electrónico y elementos con los cuales se conforma dicha planta se pueden observar en la figura 3.6.

El motor DC de 9V, disco ranurado, sensor de velocidad encoder y el servomotor SG-90, se utiliza en el módulo debido a la disponibilidad en el mercado, cumple con todos los requerimientos necesarios para elaborar el WEPC y al grupo que va dirigido está familiarizado con su uso y funcionamiento.

Elementos:

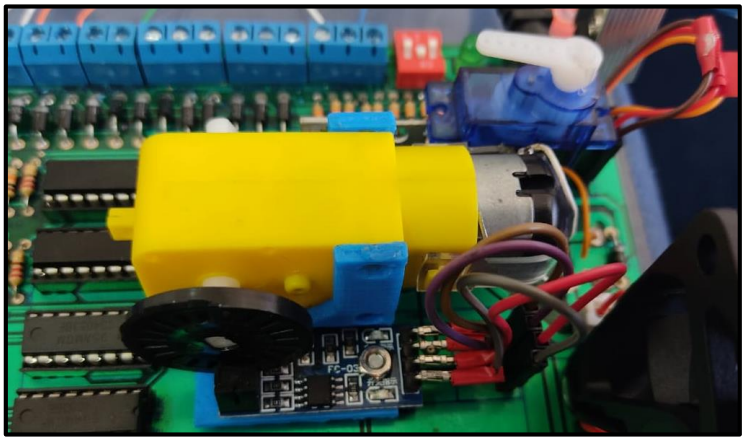
- Q3 IRFZ44N transistor activación motor
- D1 M diodo de protección transistor
- D1 S-M diodo protección señal externa del motor DC al NodeMCU
- D2 S-S diodo protección señal externa del servomotor SG-90 al NodeMCU

Figura 3. 6 Esquemático Velocidad/Posición



Elaborado por: Pablo Sunta y Diana Yánez

Figura 3. 7 Planta de Velocidad y Posición



Elaborado por: Pablo Sunta y Diana Yánez

3.1.3 Planta temperatura y nivel

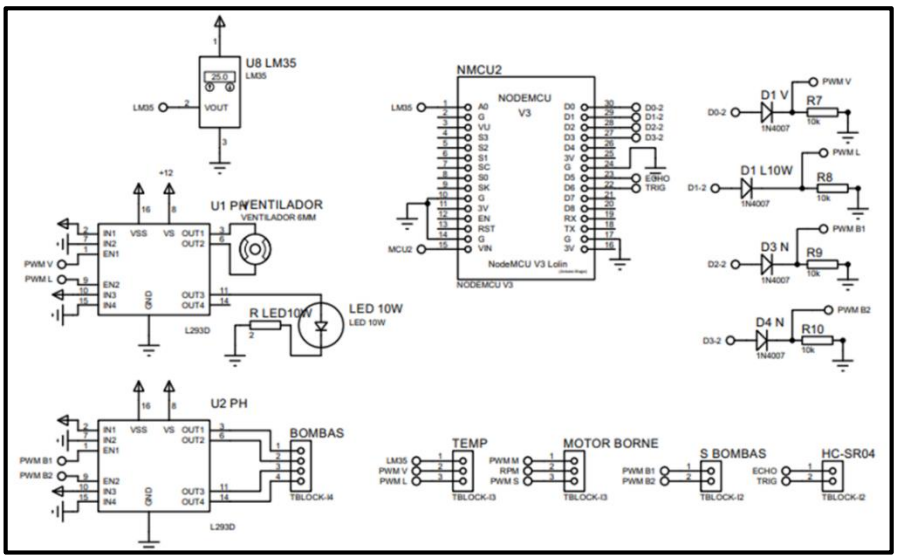
Dicha planta está conectada al segundo controlador, consta de 3 elementos un sensor LM35 el cual mide la temperatura del ambiente, un led de alta potencia de 10 [W] que maneja el aumento de la misma, el ultimo es un ventilador de trabaja como una perturbación al sistema y ejecuta el enfriamiento en la planta, juntos ejecutan las condiciones que la tarjeta embebida procese. Los elementos de esta planta son los mismos usados en el EPC de Datalights.

En la planta de nivel se diseña el circuito de tal forma que pueda conectar dos bombas de 5[V], sensor ultrasónico HC-SR04 los mismos que son externos al WEPC, los cuales cumplirían la función de llenar, evacuar y monitorear el líquido de uno de los tanques. Se usa bombas de 5V ya que la mayoría de elementos en el WEPC funcionan con ese voltaje y estas bombas son los más accesibles en cuanto a disponibilidad y economía.

Elementos:

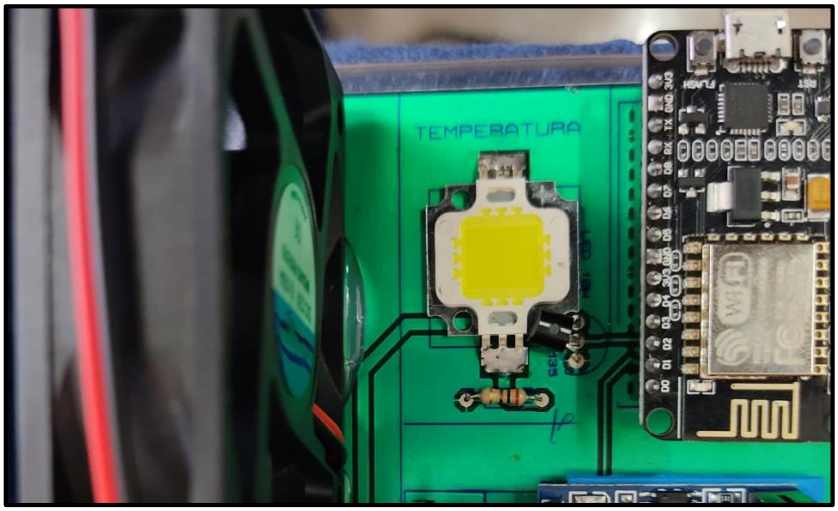
- U8 LM35 sensor de temperatura
- U1 L293D puente H control de ventilador y led 10W
- U2 L293D puente H control de bombas de agua 5v
- D1 V diodo protección señal externa del ventilador al NodeMCU
- D1 L10W diodo protección señal externa del LED 10 W al NodeMCU
- D1 V diodo protección señal externa del ventilador al NodeMCU
- D3 N, D4 N diodo protección señal externa de bomba 1, 2 al NodeMCU

Figura 3. 8 Esquemático Temperatura/Nivel



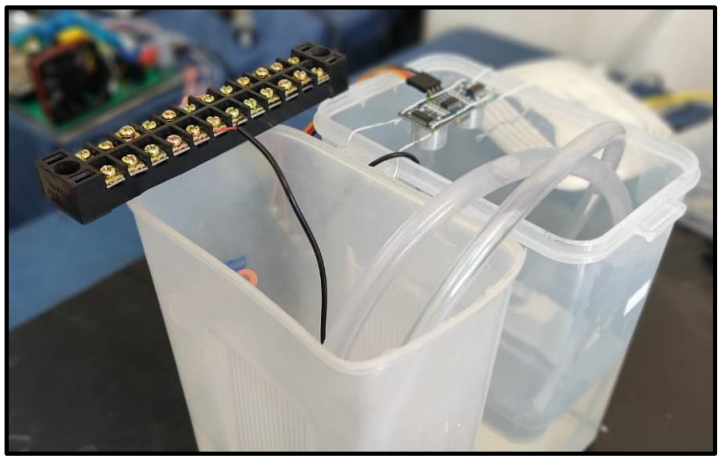
Elaborado por: Pablo Sunta y Diana Yánez

Figura 3. 9 Planta de Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

Figura 3. 10 Planta de Nivel



Elaborado por: Pablo Sunta y Diana Yánez

3.1.4 Planta de domótica

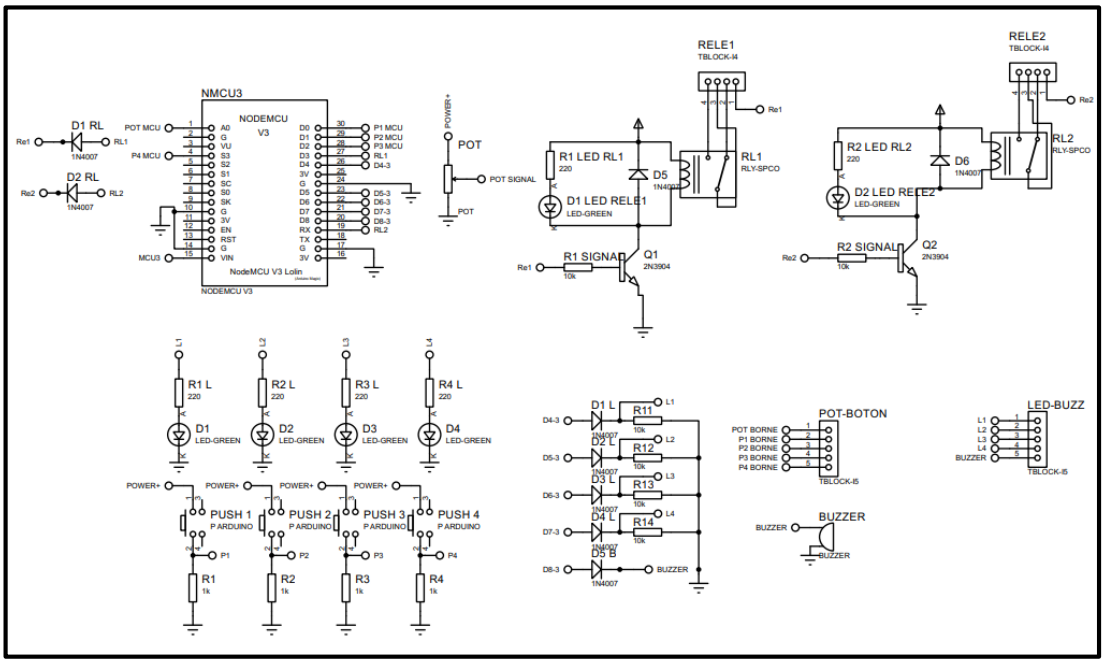
El proceso que comprende es totalmente sencillo, ejemplifica el manejo de luces que podrían ser de una casa representando así cada área del hogar, con 4 pulsadores y 4 leds, que por medio de programación pueden ser usados y controlados como sea la preferencia para el usuario, de igual forma tiene 2 relés de 5[V] de propósito general que pueden ser representados de distintas formas como por ejemplo una puerta eléctrica.

Del mismo modo tiene un potenciómetro que simula los datos de entrada de cualquier sensor y un buzzer que está representando una alarma dentro de una vivienda. Se puede observar su distribución en elementos en la figura 3.11.

Elementos:

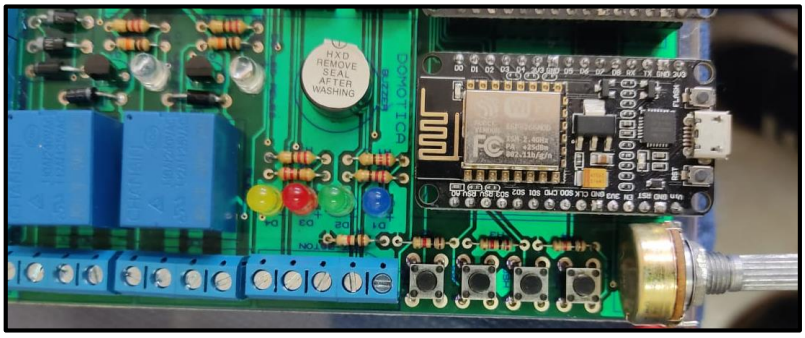
- POT potenciómetro que varía de 0v – 5v
- Buzzer actuador que emite un sonido al activar
- RL1 y RL2 relés de uso común NO
- D1, D2, D3, D4 diodo actuador emisor de luz
- PUSH 1, PUSH 2, PUSH 3, PUSH 4 botón pulsador
- D1 LED RELE1, D2 LED RELE2 led indicador del estado de RELE 1 y 2
- D5, D6 diodos de protección del transistor circuito de RELES
- D1 RL, D2 RL diodos protección señal externa de RELE 1, 2 al NodeMCU
- D1 L, D2 L, D3 L, D4 L diodos protección señal externa de Led 1, 2, 3 y 4 al NodeMCU

Figura 3. 11 Esquemático Domótica



Elaborado por: Pablo Sunta y Diana Yánez

Figura 3. 12 Planta de Domótica

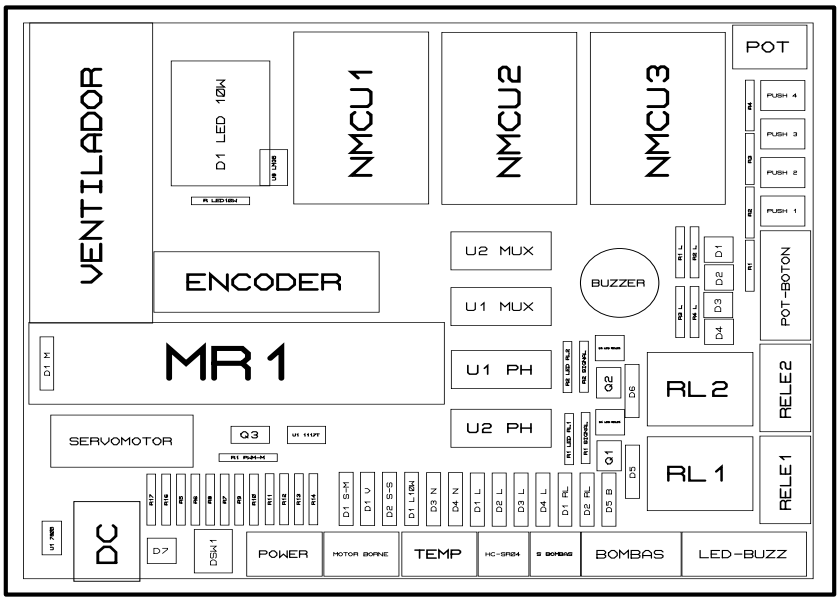


Elaborado por: Pablo Sunta y Diana Yánez

3.2 DISEÑO WEPC

El entrenador de planta de control inalámbrico es la unión de todas las plantas y controladores antes mencionados, cuenta con su respectivo sistema de alimentación y elementos electrónicos para tener un correcto funcionamiento de forma externa e inalámbrica según requiera el usuario, tiene unas dimensiones de 16 cm de largo y 13 cm de ancho, cada uno de sus elementos están distribuidos como se observa en la figura 3.13.

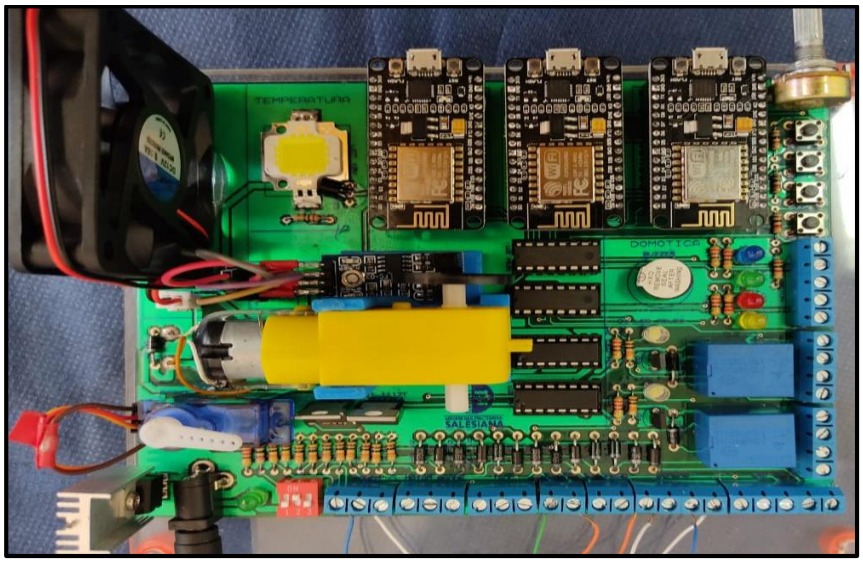
Figura 3. 13 Distribución de elementos WEPC



Elaborado por: Pablo Sunta y Diana Yánez

WEPC fue elaborada y diseñada en el software Proteus 8.9 siguiendo todos los estándares para el adecuado funcionamiento de las misma como se observa en el anexo 1, la placa PCB es de doble lado con agujeros metalizados y su respectiva serigrafía, también cuenta con mascara de soldadura para proteger cada uno de los circuitos teniendo así una placa de acabado profesional como se muestra en el anexo 2 y 3.

Figura 3. 14 WEPC

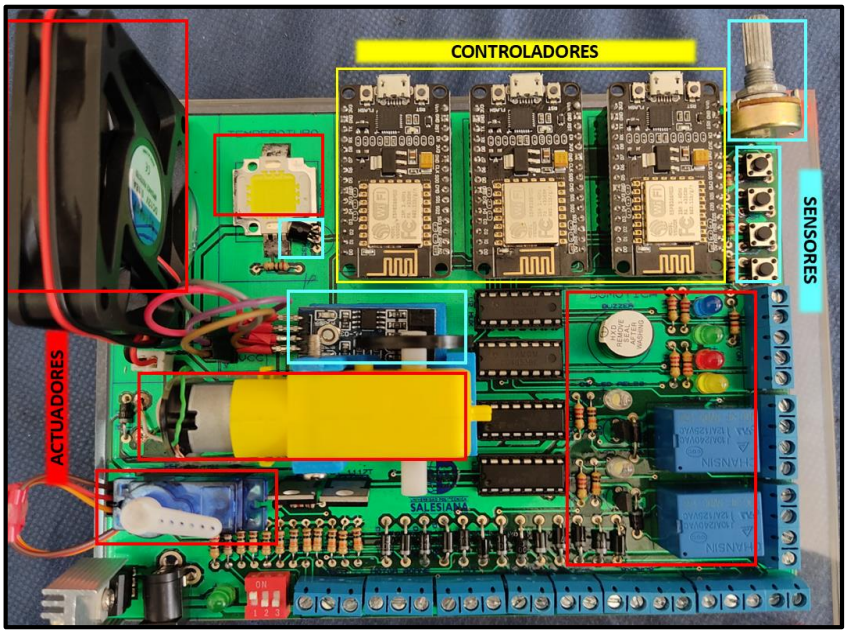


Elaborado por: Pablo Sunta y Diana Yánez

3.3 DIAGRAMA DE CONTROL DEL WNCS

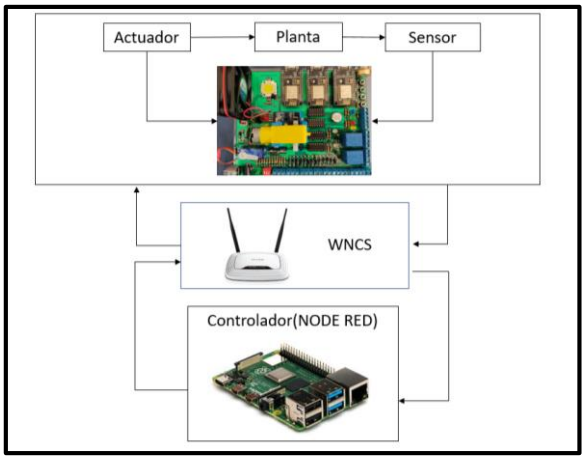
En este trabajo se ha utilizado la estructura del sistema representando la red compartida en la que el WEPC está espacialmente separado de la Raspberry PI unos de otros como se puede ver en la figura 3.15 incluyendo que la comunicación también se realiza a través de la red. Considerado que los actuadores y sensores están ubicados en el entrenador.

Figura 3. 15 Sensores y Actuadores WEPC



Elaborado por: Pablo Sunta y Diana Yánez

Figura 3. 16 Diagrama de control WNCS



Elaborado por: Pablo Sunta y Diana Yánez

El uso de la raspberry pi se fundamenta en la comparacion de la tabla 2.1 donde se encuentra las caracteristicas de las tarjeta embebidas, bajo esa comparacion y los requerimientos de conexión WiFI y procesamiento se opta por su elección añadiendo su

bajo coste, y toda la informacion que existe sobre esta.

De igual manera se elige el NodeMCU ESP8266 con respecto a su semejante el ESP32, debido a que no se necesita conexión Bluetooth ni sensores incluidos dentro del microcontrolador, ya que el objetivo es manipular dichas variables independientes al microcontrolador. Sobremanera el NodeMCU es muy facil de encontrar en el mercado y tiene un valor módico.

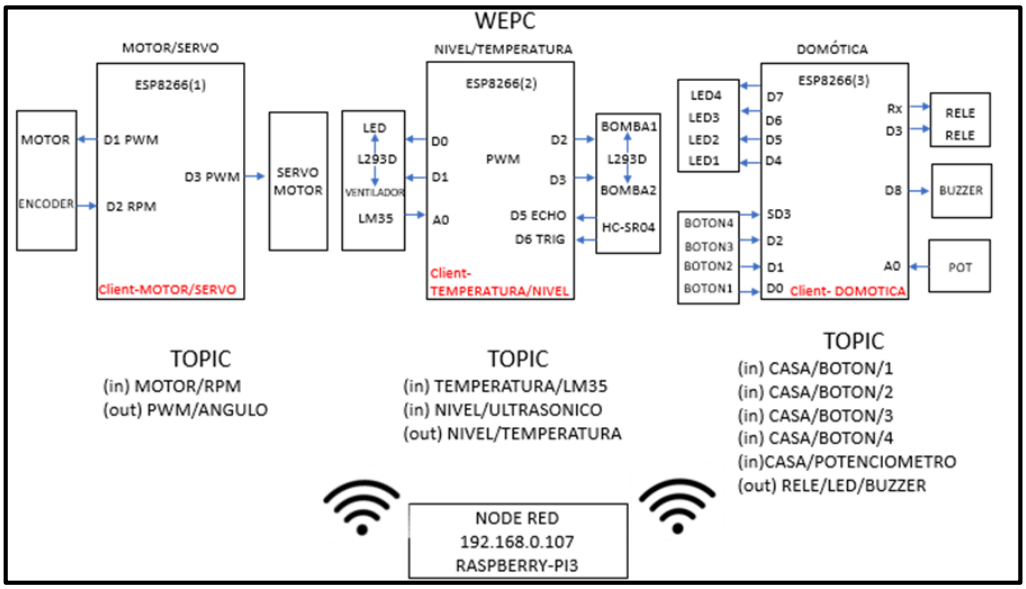
Se utiliza tres NodeMCU ESP8266 en el WEPC basado en el objetivo de entrenar al usuario, se divide todas las plantas en tres tarjetas y así aprender a administrar y manipular todos los datos y sus elementos, simulando que las plantas se encuentren remotas entre si y aplicar un correcto control inalámbrico. Sin embargo, tomando en cuenta que las ESP8266 utilizadas en la placa cuentan con más entradas y salidas no son suficientes para el control de las 5 plantas en conjunto bajo esta consideración se optó en dividir las plantas anteriormente explicadas en tres NodeMCU.

3.4 DIAGRAMA DE BLOQUES Y COMUNICACIONES EN EL WEPC

Las comunicaciones entre el WEPC y el BROKER están distribuidas de la siguiente manera la Raspberry pi 3 es el bróker o servidor el cual tiene una ip estática 192.168.107.

El WEPC tiene 3 tarjetas embebidas que es el NodeMCU ESP8266 las cuales tienen su respectivo nombre de Cliente, cada variable tiene sus propios topics de entrada o de salida con referencia a la Raspberry pi3 todos los elementos anteriormente mencionados se detallan con mayor claridad en la figura 3.17.

Figura 3. 17 Diagrama de bloques y comunicaciones WEPC



Elaborado por: Pablo Sunta y Diana Yánez

3.5 IMPLEMENTACIÓN COMUNICACIÓN MQTT ENTRE WEPC - RASPBERRY PI 3

Message Queue Telemetry Transport (MQTT) es un protocolo de máquina a máquina. A diferencia de HTTP que es protocolo de mensajería petición/respuesta, MQTT está fundamentado en publicación/suscripción. Para implementar una comunicación MQTT hay que instalar el protocolo en cada dispositivo que se vaya comunicar.

3.5.1 Comunicación Raspberry PI 3 – MQTT

La Raspberry debe tener instalado el sistema operativo "Raspbian with desktop and recommended" software, recomendable su última versión Debian Bustert.

Para instalar MQTT en una Raspberry esta debe tener una conexión a internet, se va a utilizar un servidor ampliamente conocido y de software libre como es Eclipse Mosquitto.

- Primero abrir un terminal en el Raspberry pi.
- Digitar los siguientes comandos para descargar el fichero de mosquitto.

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
```

- Se incorpora la clave de una lista para acreditar el paquete que se va a descargar posteriormente.

```
sudo apt-key add mosquitto-repo.gpg.key
```

- Acceder a la siguiente carpeta usando el comando.

```
cd /etc/apt/sources.list.d/
```

- A continuación descargar la nómina de repositorios de Mosquitto con wget, sabiendo que Buster es nuestra versión de Rasbian digitar lo siguiente.

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
```

- Para no digitar repetitivamente el código “sudo”, teclear en la terminal el siguiente comando consecuentemente ser root user.

```
sudo -i
```

- Actualizar los repositorios.

```
apt-get update
```

- Ejecutar el siguiente comando para instalar el Broker Mosquitto.

```
apt-get install mosquitto
```

- Como resultado tendrá el Broker Mosquitto funcionando en la Raspberry Pi. Como consecuencia se puede ejecutar el cliente, el cual puede ser cualquier ordenador o dispositivo en nuestra red WLAN/LAN para hacer las pruebas.

3.5.2 Comunicación NodeMCU ESP8266 – MQTT

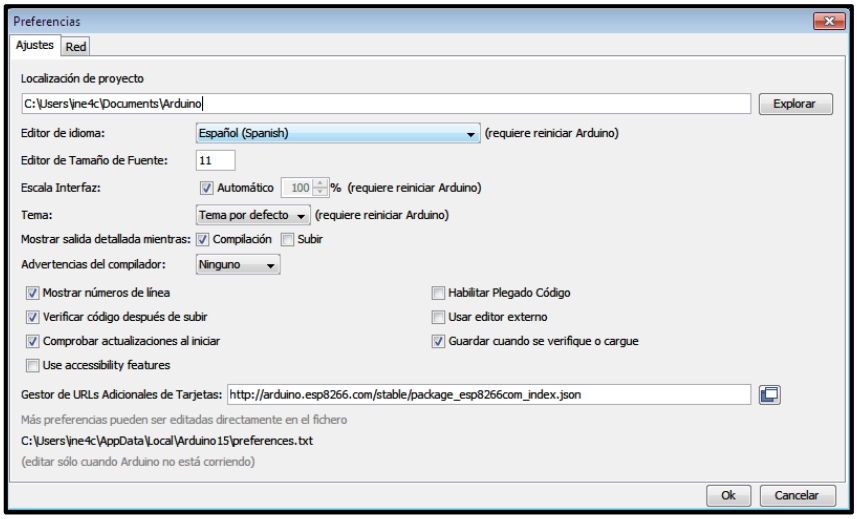
3.5.2.1 Instalar pluggin ESP9266 en el IDE

Se necesita tener la versión 1.6.4v o superior del IDE de Arduino, para instalar el pluggin hay que tener una conexión a internet.

- Abrir el entorno de Arduino y seleccionar en Archivo > Preferencias

- En la parte inferior en Gestor de URLs Adicionales de Tarjetas, y escribir ahí: http://arduino.esp8266.com/stable/package_esp8266com_index.json

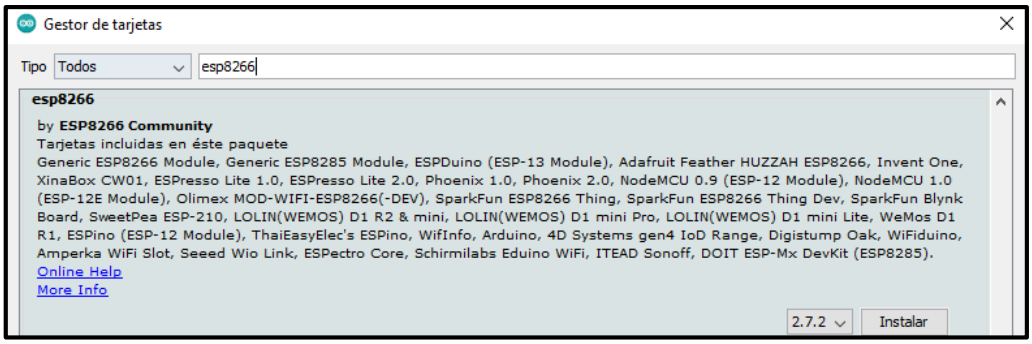
Figura 3. 18 Ventana de instalación pluggin



Elaborado por: Pablo Sunta y Diana Yánez

- Seleccionar OK
- Abrir la opción del menú Herramientas > Placa > Gestor de tarjetas buscar la opción ESP8266 Comunity, seleccionar su última versión e instalarlo

Figura 3. 19 Ventana de instalación tarjeta ESP8266



Elaborado por: Pablo Sunta y Diana Yánez

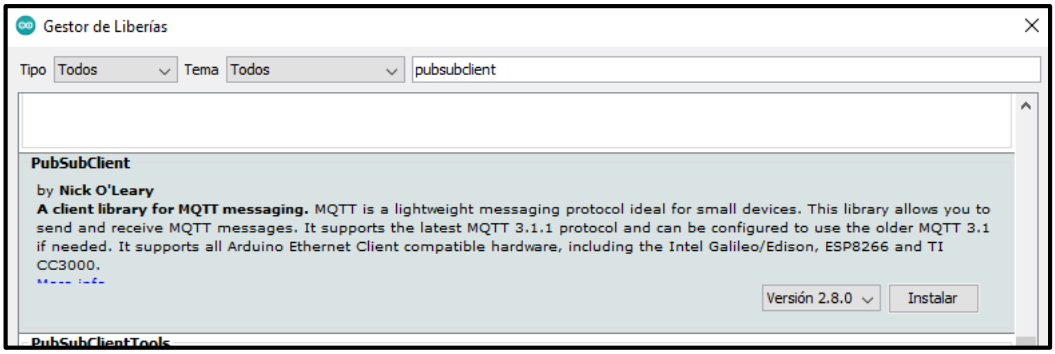
- Al dirigirse a \Herramientas\Placas se observa que ya se puede seleccionar todas las placas correspondientes al ESP8266.
- La comunicación con el módulo y el IDE lo vamos a hacer por el puerto serial, preferiblemente usar una velocidad a 115.200. Como resultado ya está en su estado ideal para programar el módulo WIFI.

3.5.2.2 Instalación de drivers y librerías en el IDE para utilizar el NodeMCU

Para comunicar el ESP 8266 con el protocolo MQTT necesitamos la librería PubSubClient la cual es compatible entre IDE de Arduino y NodeMCU. En definitiva, hace que nuestro modulo trabaje como un cliente, en otras palabras, lograr publicar mensajes y suscribirse a uno o varios topics para receptar mensajes.

- Seleccionar en el IDE el menú Programa > Incluir Librería > Gestor de librerías *y* buscar PubSubClient.

Figura 3. 20 Ventana de instalación librería



Elaborado por: Pablo Sunta y Diana Yánez

3.5.3 Instalar Node-RED

Para Instalar Node-RED en una Raspberry Pi hay que seguir las siguientes instrucciones:

- Abrir un terminal y digitar lo siguiente.

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-
          package/master/resources/update-nodejs-and-nodered)
```

- Este comando actualiza **NodeJS y Node-RED.** Como se mencionó antes, es importante siempre trabajar con las últimas versiones en este caso Raspbian Buster.

- Al introducir el comando preguntará "¿Estás seguro de hacer esto? [y/n]". Se Debe escribir "y" o "Y" y presionar la tecla Enter. Es preferible tener una conexión a internet mediante cable ethernet.

Una vez finalizado el Node-RED está listo trabajar. Es necesario que el Node-Red arranque automáticamente al momento de encender la Raspberry por lo tanto ejecutamos en el terminal lo siguiente.

```
sudo systemctl enable nodered.service
```

Para tener una conexión estable y sin problemas de conexión hay que asignar una IP fija a la Raspberry desde el Raspbian o desde el router por medio de la MAC de la Raspberry Pi.

3.6 PROGRAMACIÓN EN NODE-RED

A continuación, se detallará los elementos de programación con su respectivo Dashboard en NODE-RED incluyendo los topics de cada planta y explicando a que variables pertenece,

3.6.1 Planta Motor - Servomotor

La planta Motor – Servomotor está controlado por un NodeMCU el cual tiene dos actuadores un motor reductor DC y un servomotor también cuenta con un sensor óptico para detectar las RPM del motor DC a continuación se presenta los topics de cada variable y el nombre del Cliente.

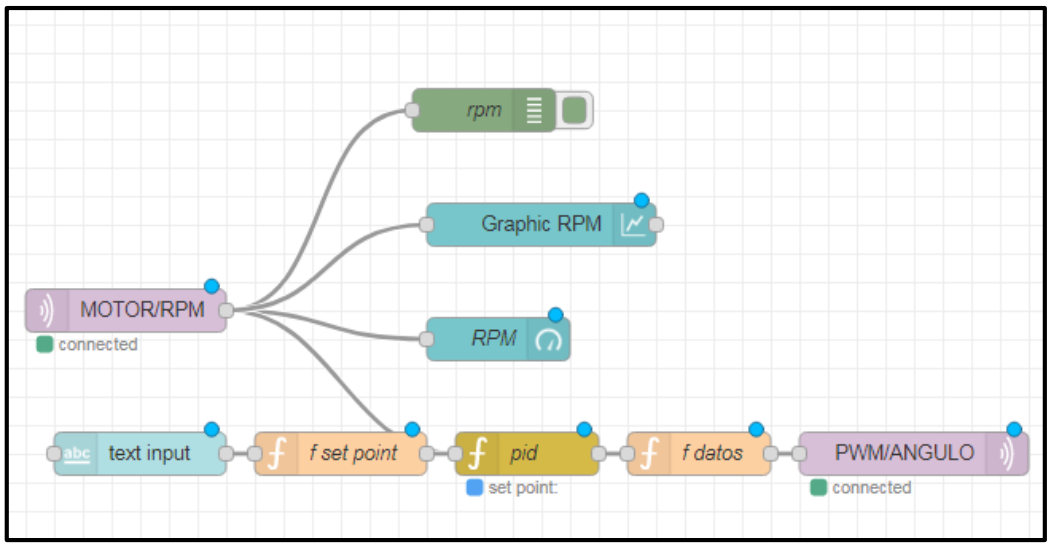
Tabla 3. 1 Topics y Variables Motor-Servomotor

Planta Motor-Servomotor		
Client: Motor/Servo		
Topic (in)	MOTOR/RPM	La velocidad del Motor en RPM
Topic (out)	PWM/ANGULO	El valor de PWM del motor entre 0-1024 acompañado del carácter “A” ejemplo: “A1024” El valor del ángulo del servomotor entre 0-180 acompañado del carácter “B” ejemplo: “B1024”

Fuente: Pablo Sunta y Diana Yánez

En el anexo 4, se encuentra la programación del NODEMCU para la planta Velocidad/Posicion. En la figura 3.21 se aprecia el algoritmo de control PID del motor DC. Los recuadros de color lila son elementos MQTT in/out: MOTOR/RPM es el nodo que recibe los datos del WEPC en rpms; PWM/ANGULO es el nodo que envia datos al WEPC. A su vez los recuadros de color azul y celeste son los nodos que estan destinados a la interfaz grafica Dashboard; y por ultimo los nodos de color amarillo son funciones que se necesitan para poder interconectar los datos entre cada uno de los nodos.

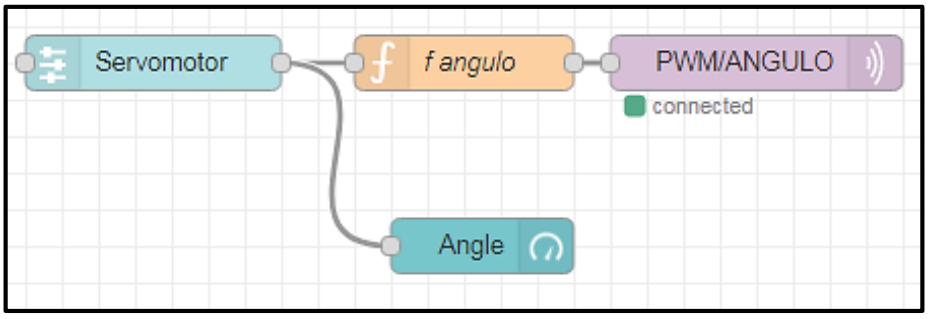
Figura 3. 21 Control PID en NODE-RED



Elaborado por: Pablo Sunta y Diana Yánez

La siguiente programación es para poder controlar el servomotor por medio de un slider y poder colocar el ángulo requerido se aplica una función necesaria para la interconexión de datos y su respectivo MQTT out: PWM/ANGULO.

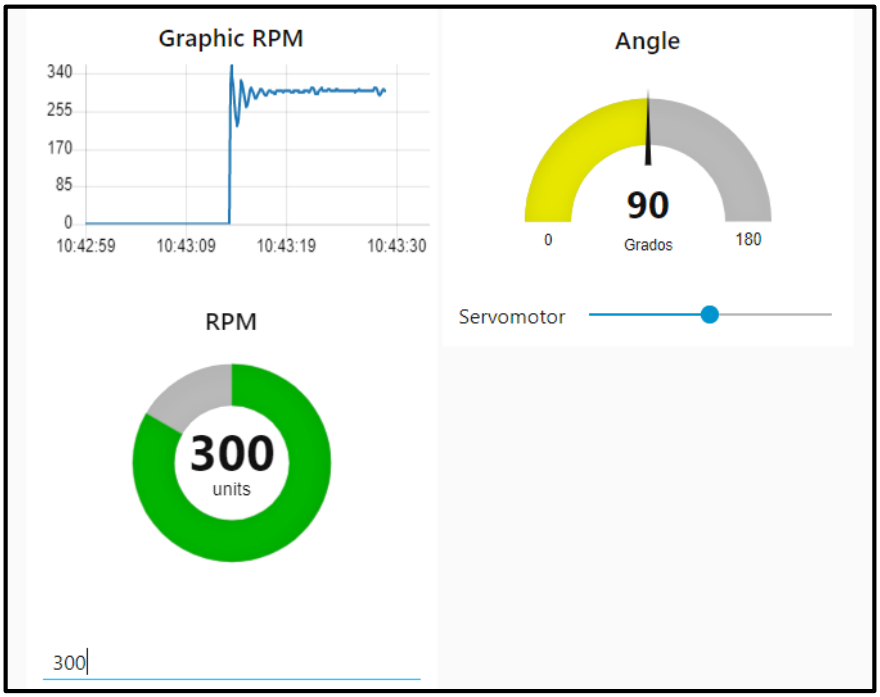
Figura 3. 22 Programación Servomotor NODE-RED



Elaborado por: Pablo Sunta y Diana Yánez

El Dashboard Planta Motor DC consta de 3 partes; de un gráfico de tipo Char para poder observar el comportamiento de RPM del motor a lo largo del tiempo; gráfico Gauge de tipo donout para observar la variable RPM en tiempo real; un elemento text input para poder introducir el set point solicitado por el usuario. La planta Servomotor consta de dos partes; un slider limitado entre los valores de 0 y 180; y un gráfico tipo Gauge para observar el ángulo aplicado en el servomotor.

Figura 3. 23 Motor DC y Servomotor



Elaborado por: Pablo Sunta y Diana Yánez

3.6.2 Planta de temperatura y nivel

La planta Temperatura - Nivel está controlado por un NodeMCU el cual tiene cuatro actuadores: un led de 10w un ventilador y dos bombas de agua DC también cuenta con un sensor ultrasónico el cual envía los datos del nivel de agua que se encuentran los tanques a continuación, se presenta los topics de cada variable y el nombre del Cliente.

Tabla 3. 2 Topics y Variables Temperatura- Nivel

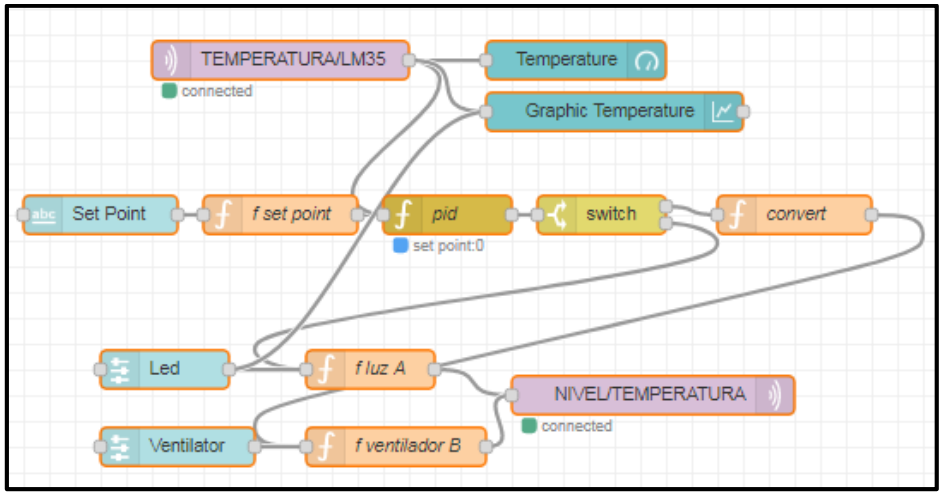
Planta: Temperatura - Nivel		
Client: NIVEL/TEMPERATURA		
Topic (in)	TEMPERATURA/LM35	La temperatura del LM35 del WEPC en grados centígrados
Topic (in)	NIVEL/ULTRASONICO	El nivel del tanque de agua medido por el sensor ultrasónico HC SR04 en centímetros
Topic (out)	NIVEL/TEMPERATURA	El valor de PWM del led para aplicar calor en el WEPC entre 0-1024 acompañado del carácter "A" ejemplo: "A1024" El valor de PWM del ventilador para enfriar el WEPC entre 0-1024 acompañado del carácter "B" ejemplo: "B1024" El valor de PWM de la primera bomba de agua para ingresar liquido al tanque principal del WEPC entre 0-1024 acompañado del carácter "C" ejemplo: "C1024" El valor de PWM de la segunda bomba de agua encargada de sacar liquido del tanque principal del WEPC entre 0-1024 acompañado del carácter "D" ejemplo: "D1024"

Fuente: Pablo Sunta y Diana Yánez

En el anexo 4, se encuentra la programación del NODEMCU, para la planta de Temperatura/Nivel. Como se aprecia en la figura 3.24 se observa el algoritmo de control PID de temperatura del WEPC. Los recuadros de color lila son elementos MQTT in/out: TEMPERATURA/LM35 es el nodo que recibe los datos de temperatura en °C; NIVEL/TEMPERATURA es el nodo que envia datos al WEPC. A su vez los recuadros de color azul y celeste son los nodos que estan destinados a la interfaz grafica Dashboard;

y por ultimo los nodos de color amarillo son funciones que se necesitan para poder interconectar los datos entre cada uno de los nodos

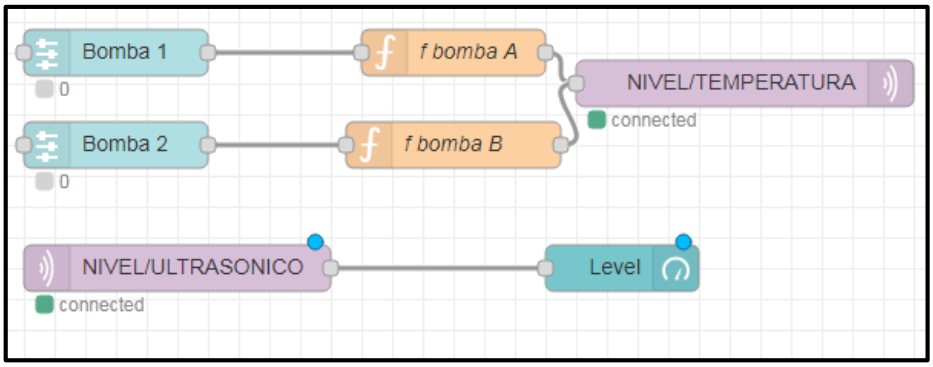
Figura 3. 24 Programación PID temperatura en NODE-RED



Elaborado por: Pablo Sunta y Diana Yánez

En la programación de la planta de nivel tenemos dos Slider para controlar el pwm en cada una de las bombas seguidas de sus respectivas funciones y de la misma manera que en la figura 3.25 tenemos los MQTT out/in para recibir datos y enviar órdenes a cada cliente.

Figura 3. 25 Programación Nivel/Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

El Dashboard planta temperatura DC consta de cuatro partes; de un gráfico de tipo Char para poder observar el comportamiento de la temperatura en °C a lo largo del tiempo; gráfico de tipo donout para observar la variable temperatura en tiempo real; un elemento text input para poder introducir el set point; dos slider para comprobar el funcionamiento del led y del ventilador.

La planta nivel consta de dos partes; dos slider limitados entre los valores de 0 y 1024; y un gráfico tipo Level para observar el nivel de agua del recipiente en cm.

3.6.3 Planta de domótica

La planta Domótica está controlada por un NodeMCU el cual tiene siete actuadores como son cuatro leds, dos relés, un buzzer también cuenta con elementos de entrada como son cuatro botones y un potenciómetro, a continuación, se presenta los topics de cada variable y el nombre del Cliente.

Tabla 3. 3 Topics y Variables Domótica

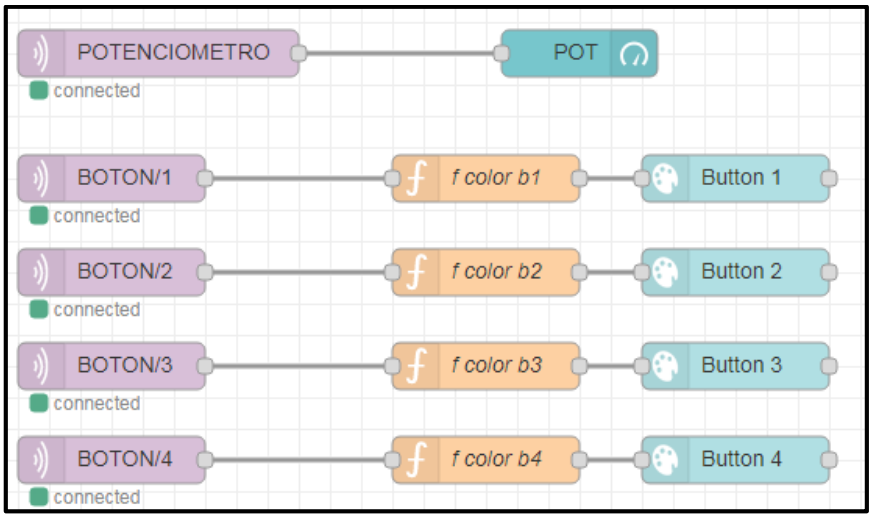
Planta: Domótica		
Client: Domotica		
Topic (in)	BOTON/1	El estado on/off del botón 1 del WEPC
Topic (in)	BOTON/2	El estado on/off del botón 2 del WEPC
Topic (in)	BOTON/3	El estado on/off del botón 3 del WEPC
Topic (in)	BOTON/4	El estado on/off del botón 4 del WEPC
Topic (in)	POTENCIOMETRO	El valor analógico del potenciómetro limitado entre valores de 0-1024
Topic (out)	RELE/LED/BUZZER	El estado del led 1 ON=1, OFF=0 acompañado del carácter “A” ejemplo: “A1” El estado del led 2 ON=1, OFF=0 acompañado del carácter “B” ejemplo: “B1” El estado del led 3 ON=1, OFF=0 acompañado del carácter “C” ejemplo: “C1” El estado del led 4 ON=1, OFF=0 acompañado del carácter “D” ejemplo: “D1”

		El estado del Relé 1 ON=1, OFF=0 acompañado del carácter “E” ejemplo: “E1” El estado del Relé 2 ON=1, OFF=0 acompañado del carácter “D” ejemplo: “D1” El estado del Buzzer ON=1, OFF=0 acompañado del carácter “F” ejemplo: “F1”

Elaborado por: Pablo Sunta y Diana Yánez

En la programación vamos a dividir en dos partes elementos de entrada y elementos de salida. En los elementos de entrada leemos con un MQTT in las variables antes detalladas en la tabla 3. 3, a cada topic le asignamos su función correspondiente para la comunicación entre nodos y su respectivo indicador.

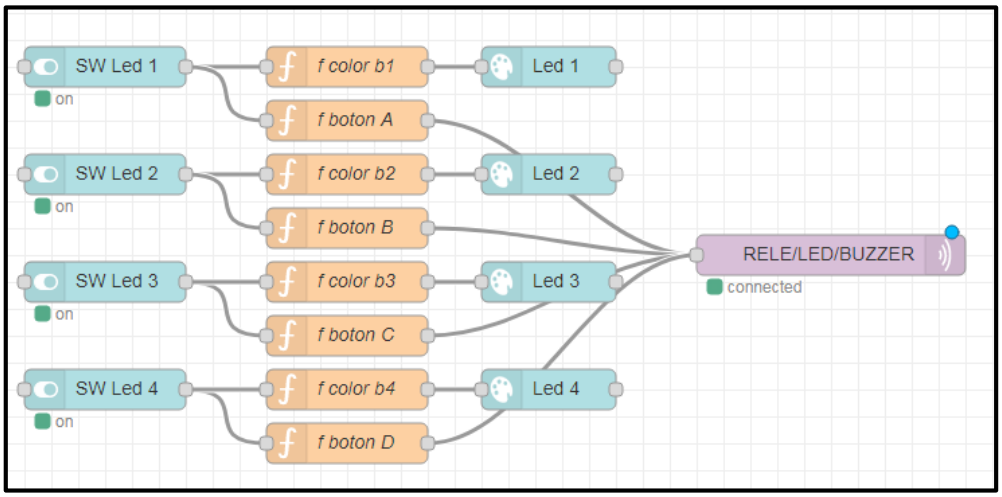
Figura 3. 26 Programación Entradas domótica



Elaborado por: Pablo Sunta y Diana Yánez

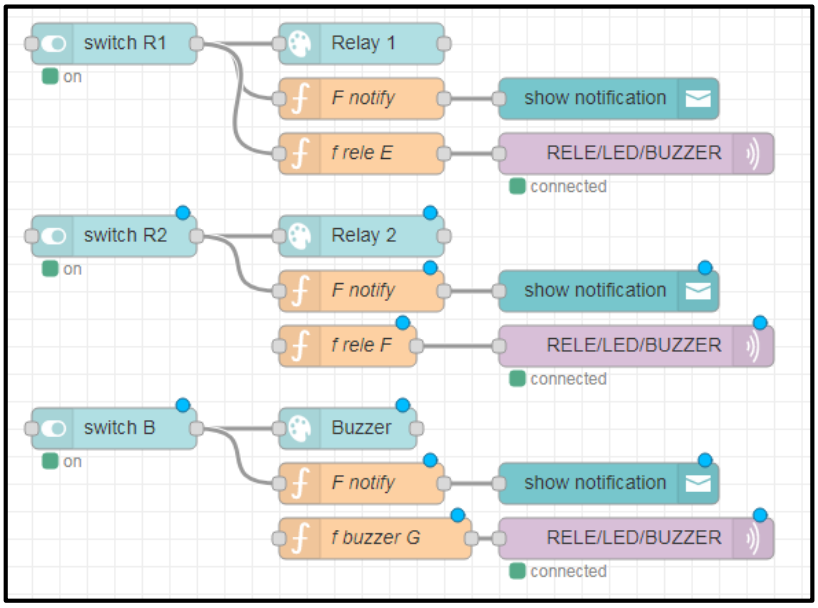
En los elementos de salida, asignamos un switch virtual para poder forzar el estado de cada uno de siete los actuadores en este caso leds, relés, buzzer; se le asigna su función correspondiente para poder enlazar datos con el indicador en el dashboard y su envió de datos al WEPC con MQTT OUT.

Figura 3. 27 Programación Salidas domótica leds



Elaborado por: Pablo Sunta y Diana Yánez

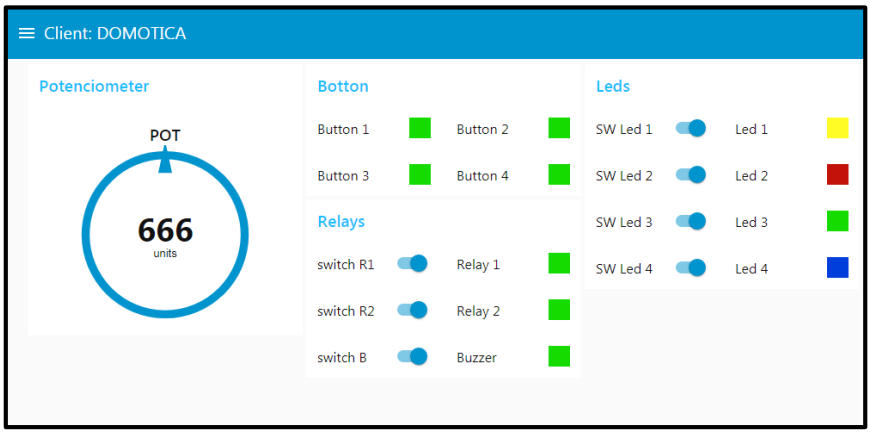
Figura 3. 28 Programación Salidas domótica relé/buzzer



Elaborado por: Pablo Sunta y Diana Yánez

El dashboard de la planta de domótica consta de 4 partes: un gráfico tipo Compass para indicar el valor que nos envía el potenciómetro; la sección de bottons con su respectivos indicadores pinta de color blanco cuando esta “off” y color verde cuando está en “on”; la sección de Leds tiene sus respectivos switchs virtuales y sus indicadores los cuales pintan del color de los leds del WEPC; la sección de relés y Buzzer de igual manera tiene switch virtuales para forzar aquellos actuadores, cuenta de indicadores los cuales pintan de color blanco en “off” y verde en “on”, cuando un relé cambia de estado surge una notificación en pantalla con todos sus detalles.

Figura 3. 29 Dashboard domótica



Elaborado por: Pablo Sunta y Diana Yánez

CAPÍTULO 4
PRUEBAS Y RESULTADOS

Este capítulo detalla las pruebas que se realizaron para obtener las funciones de transferencia de cada planta, puesto que esta es un elemento importante que permite estudiar el comportamiento de determinado proceso, bien sea académico o industrial, a en el transcurso del tiempo y a su vez se aplicó un algoritmo de PID en la planta de Motor DC y temperatura para comprobar su funcionamiento.

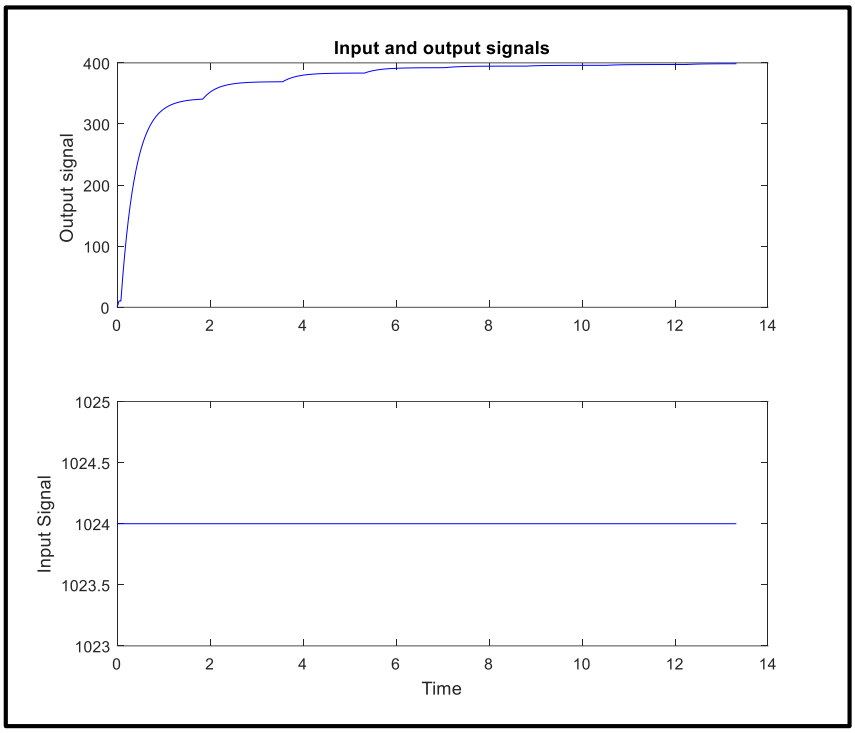
4.1 FUNCIÓN DE TRANSFERENCIA PLANTA MOTOR DC

En el proyecto se logra obtener de forma experimental el comportamiento de un motor DC de 3[V] a 12[V] a lo largo del tiempo, con el objetivo de adquirir su modelo matemático que proporcione saber la respuesta de dicha planta ante distintos valores de entrada.

Para constituir el sistema y los factores que describen al motor matemáticamente, se aplica un método de adquisición de datos por medio del puerto serial en el IDE Arduino y usando el WEPC, se acciona el motor con una señal de entrada PWM de 1024 y por medio el sensor óptico, encoder se obtiene la señal de salida en revoluciones por minuto [rpm] a las que gira el motor.

Para adquirir la respuesta de la señal, se toman los datos por el puerto serial cada 40 [mS], se los guarda en una hoja de cálculo y se los envía al software MATLAB, el mismo que posee la herramienta IDENT para obtener la función de transferencia con un método más rápido y seguro.

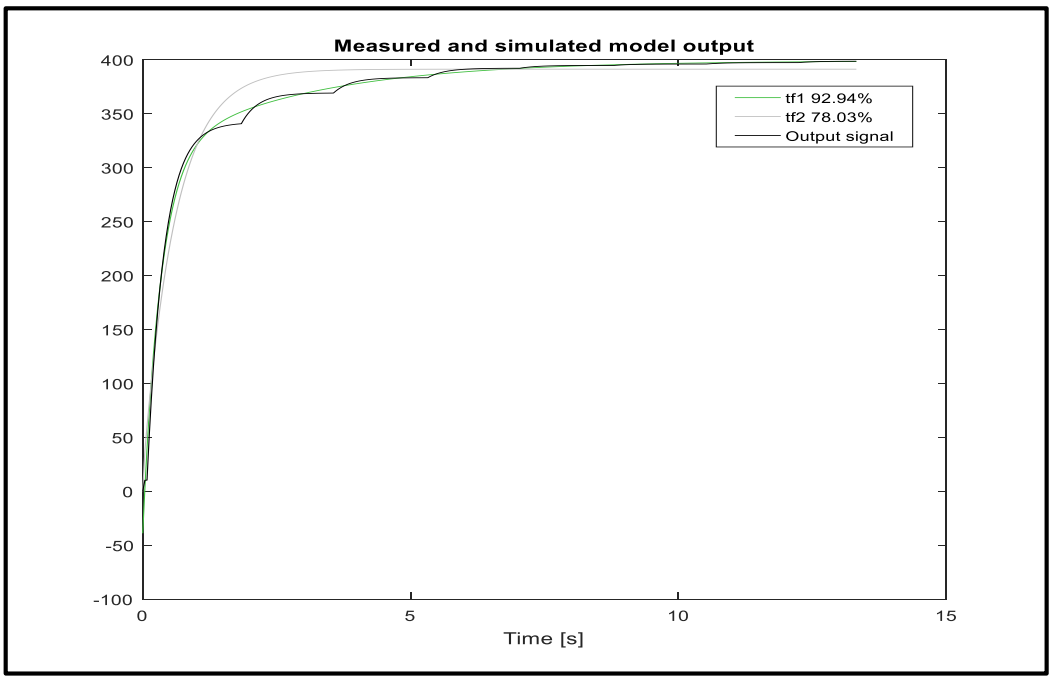
Figura 4. 1 Datos y Set Point Planta Motor DC



Elaborado por: Pablo Sunta y Diana Yánez

En la herramienta IDENT de Matlab se obtuvo dos funciones de transferencia la primera (tf1) con dos polos y un cero, la segunda (tf2) con dos polos y ningún cero.

Figura 4. 2 Función de transferencia Planta de Motor DC



Elaborado por: Pablo Sunta y Diana Yánez

$$tf1 = \frac{0.9477\ s + 0.4272}{s^2 + 3.419\ s + 1.096} \qquad Ec(1)$$

$$tf2 = \frac{103.2}{s^2 + 164.1\ s + 270.1} \qquad Ec(2)$$

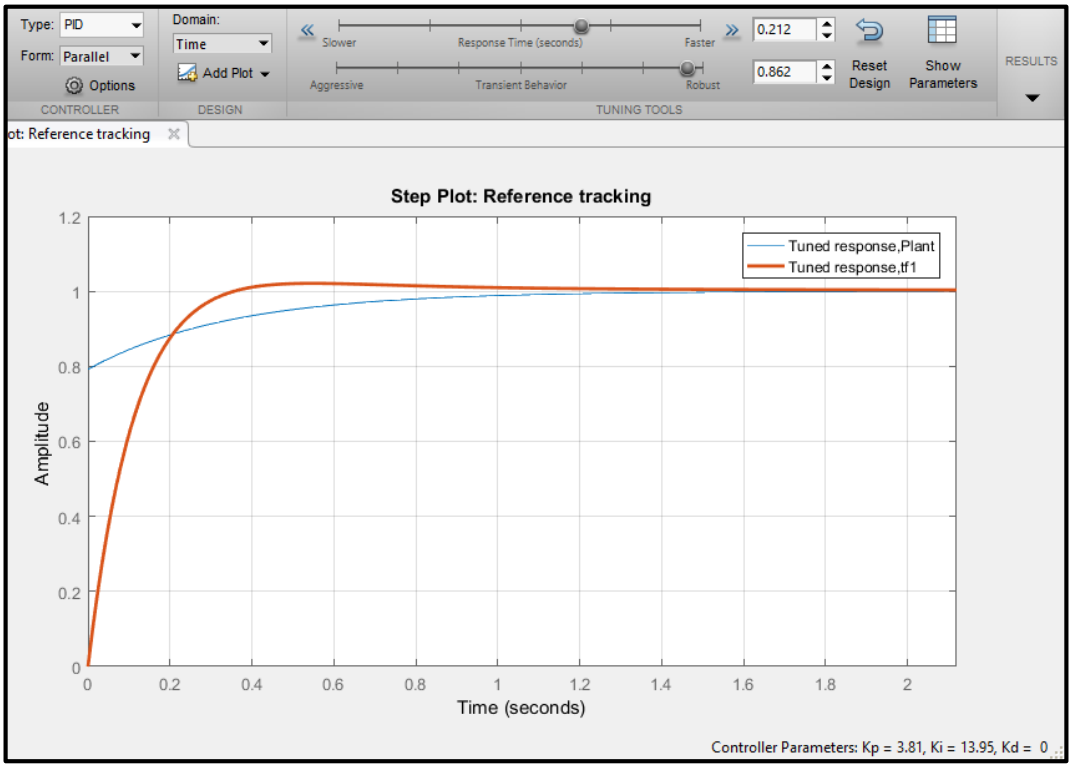
Como se observa en la figura 4.2 la primera función de transferencia tf1 es la más cercana a los datos originales con un 92.94% de aproximación, y tf2 es tiene 78.3% de aproximación.

4.1.1 PID planta de Velocidad

Para aplicar un algoritmo de control PID en la planta de temperatura se utilizó una librería de Node-Red llamada "node-red-node-pidcontrol" la cual procesa los datos de set point y valores del sensor en este caso es el nodo mqtt in con el topic "MOTOR/RPM" y así dar una respuesta adecuada al actuador que se está comunicando con mqtt out con el topic "PWM/ANGULO".

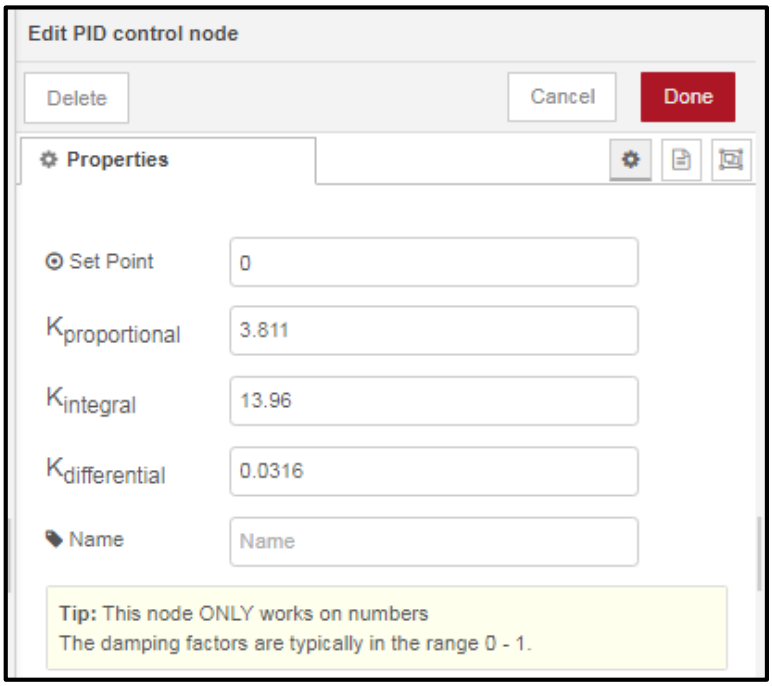
Los valores del PID fueron obtenidos en la herramienta del Matlab PIDTOOL, y ensayados en el WEPC observando así su comportamiento.

Figura 4. 3 PID Tuner Matlab para Planta de Velocidad (Motor DC)



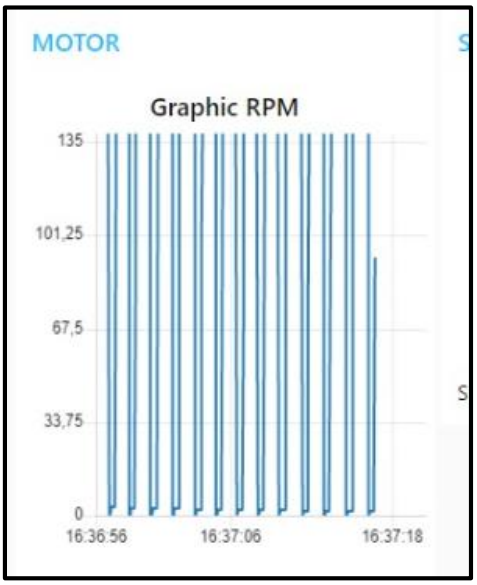
Elaborado por: Pablo Sunta y Diana Yánez

Figura 4. 4 Valores Kp, Ki, Kd según PID Tuner



Elaborado por: Pablo Sunta y Diana Yánez

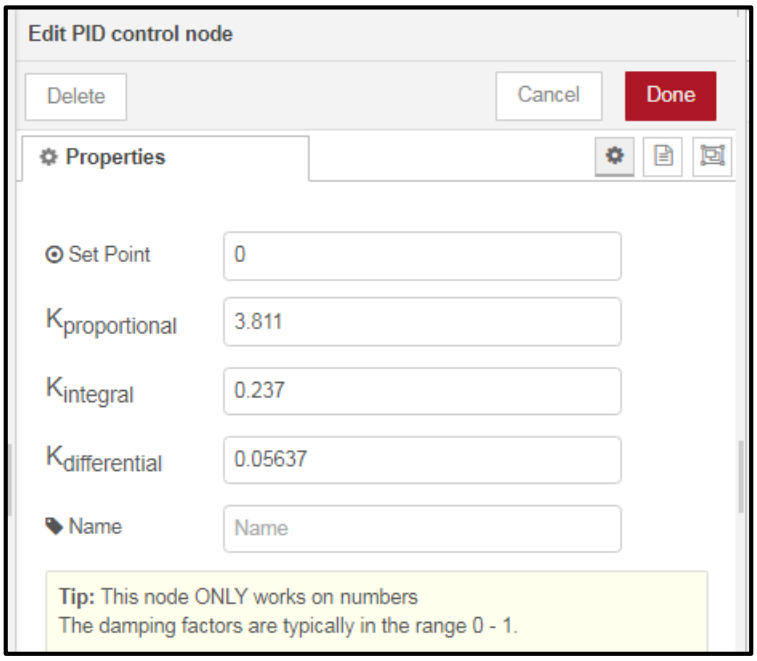
Figura 4. 5 Gráfica Resultante



Elaborado por: Pablo Sunta y Diana Yánez

En la planta de motor DC se aplicó estos valores de Kp, Ki, Kd viendo así que la planta se comporta como un sistema inestable, con ese antecedente se va cambiando los valores de Ki y Kd según el comportamiento de la planta. Una vez realizado la experimentación se logró estabilizar el sistema con los siguientes valores Kp= 3.811, Ki= 0.237, Kd=0.05637 como se observa en la figura 4.6 con un set point de 105 [rpm].

Figura 4. 6 Kp, Ki, Kd finales



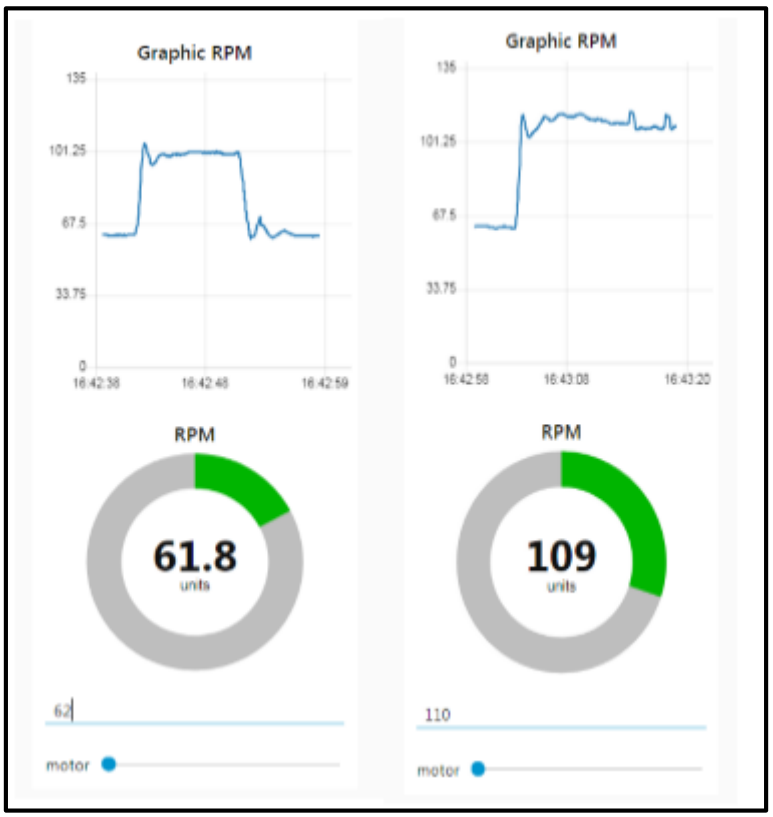
Elaborado por: Pablo Sunta y Diana Yánez

Figura 4. 7 Gráficas y PID resultante Motor PID

Elaborado por: Pablo Sunta y Diana Yánez

Se coloca diferentes set point para comprobar el funcionamiento del PID el primer valor es de 101 [rpm] el segundo valor es de 62 [rpm] y por último se coloca un tercer set point de 110 [rpm] así como indica la figura 4.8.

Figura 4. 8 Set point de 101 rpm ,62 rpm y 110 rpm



Elaborado por: Pablo Sunta y Diana Yánez

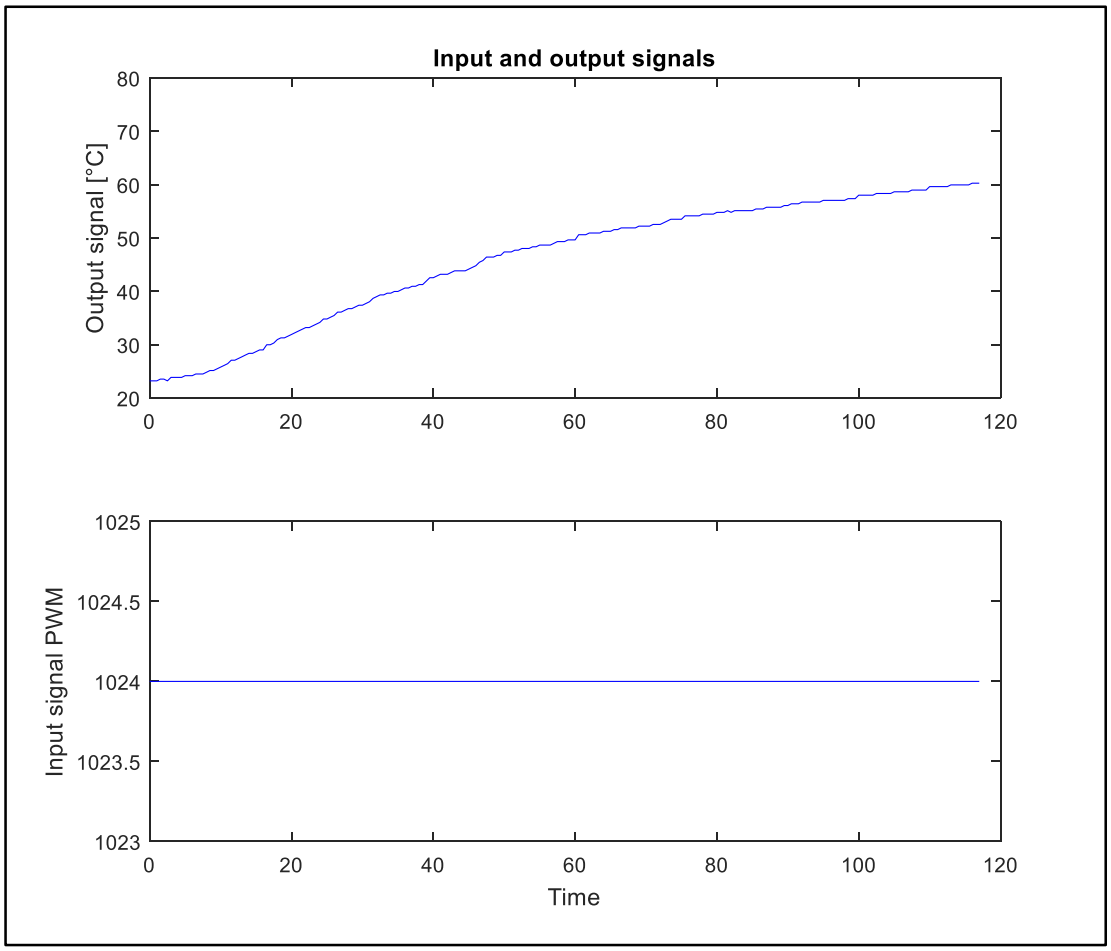
4.2 FUNCIÓN DE TRANSFERENCIA PLANTA DE TEMPERATURA

La temperatura es muy importante en varios sistemas de control ya que muchas veces puede afectar directamente a otras variables sea en producción u otros procesos especialmente en espacios industriales. El WEPC tiene una planta de temperatura con dos actuadores: un led de 10 [W] el cual manipulará todo el proceso de temperatura y un ventilador de 12[V] que servirá como una perturbación al sistema, los mismo que están listos para la manipulación del usuario.

Para constituir el sistema y los elementos que describen matemáticamente a la temperatura, se aplica un método de adquisición de datos con Arduino y usando el WEPC en forma externa, se aplica una señal de entrada al led con un valor PWM de 1024 para el calentamiento, y se coloca un set point de 60 [°C] el cual se obtiene gracias al sensor

LM35, los datos son recopilados por Arduino el cual procesa las señales eléctricas, imprime por el puerto serial cada 500 [mS] la señal de salida en grados centígrados [°C].

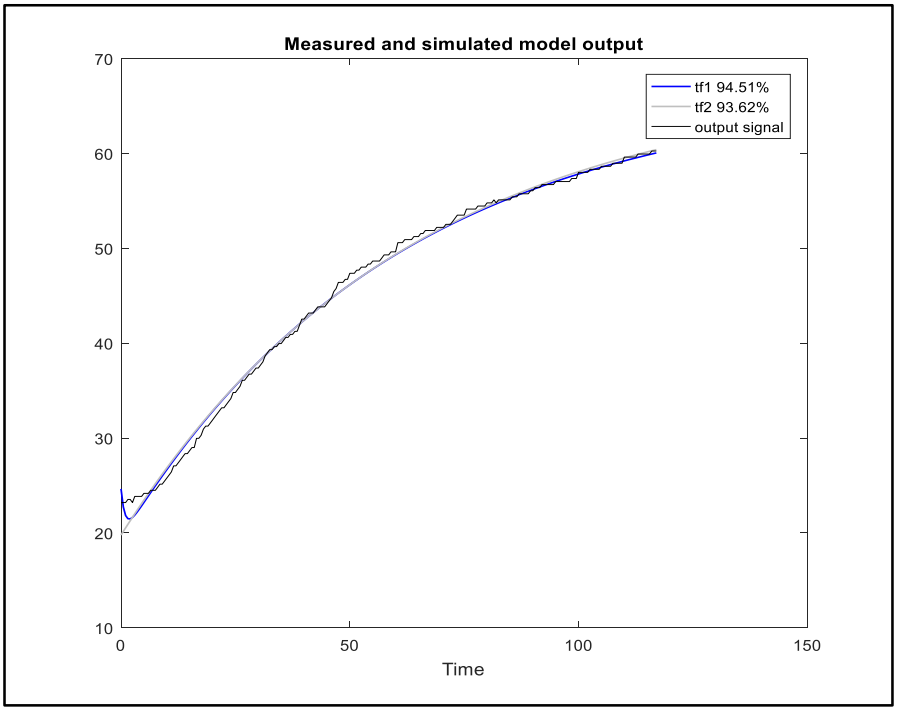
Figura 4. 9 Datos y Set Point Planta de Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

Para obtener la respuesta de la señal, se toman los datos se los almacena en una hoja de cálculo y se importan los datos al IDENT de Matlab para obtener, las gráficas y sus respectivas funciones de transferencia. De la planta de temperatura se obtuvo dos funciones de transferencia la primera (tf1) con dos polos y un cero, la segunda solo un polo, (tf2).

Figura 4. 9 Función de Transferencia Planta de Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

$$tf1 = \frac{0.04896\ s + 0.001259}{s^2 + 1.179s + 0.01927} \qquad Ec(3)$$

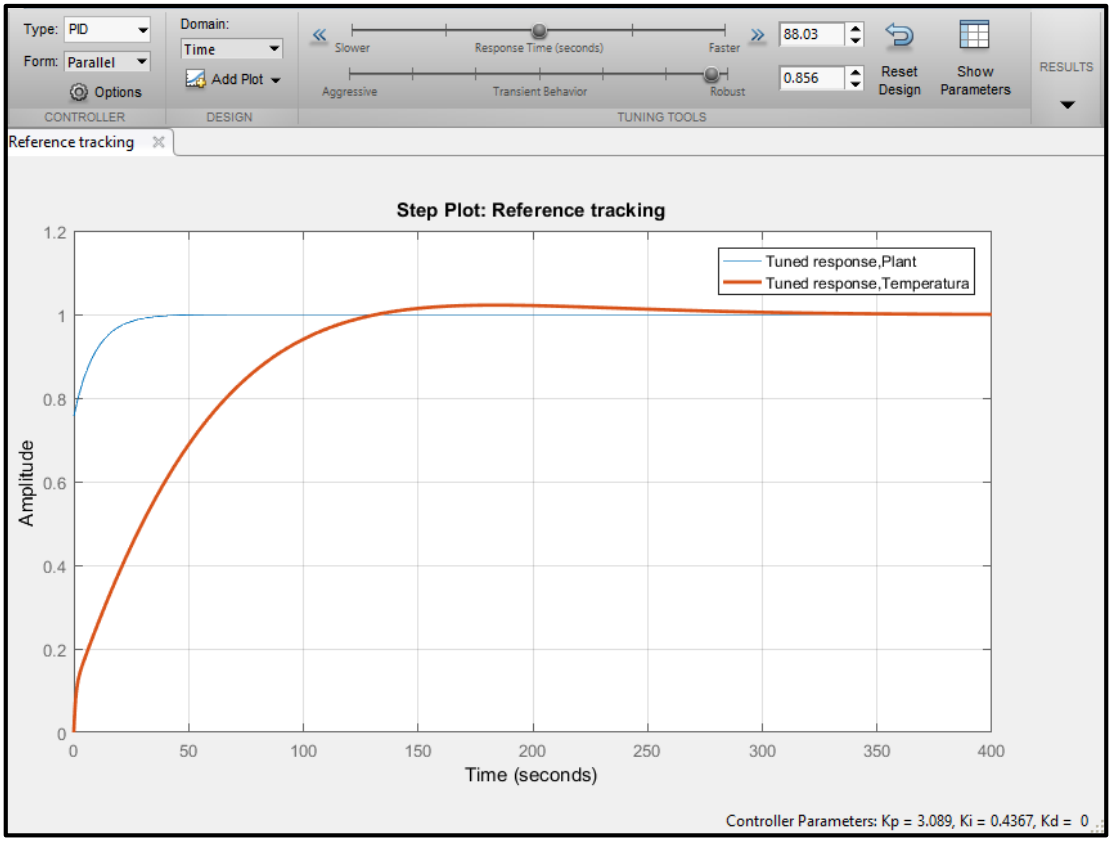
$$tf2 = \frac{0.001054}{0.08355\ s + 0.0159} \qquad Ec(4)$$

Como se observa en la figura 4.10 la primera función de transferencia tf1 tiene un 94.51% de aproximación y tf2 es tiene 93.62%, siendo así tf1 la función de transferencia más aproximada.

4.2.1 PID planta de Temperatura

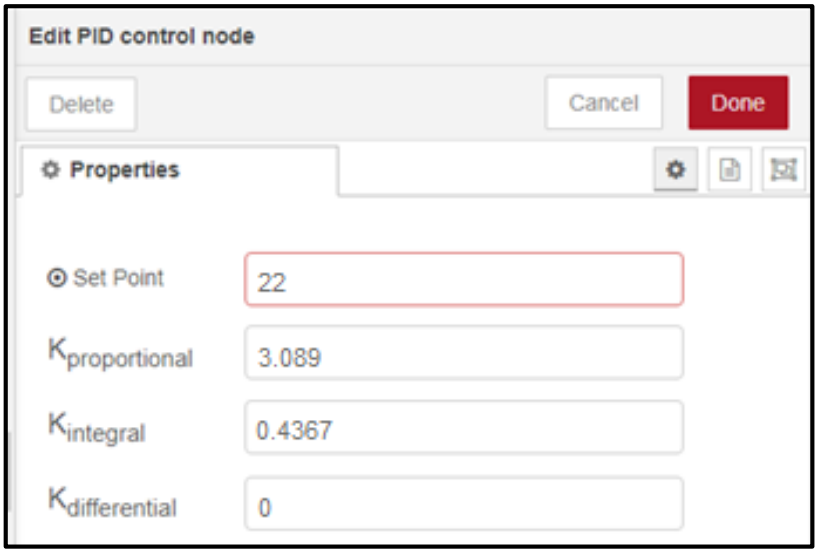
Para aplicar un algoritmo de control PID en la planta de temperatura se utilizó una librería de Node-Red llamada "node-red-node-pidcontrol" el cual procesa los datos de set point y valores del sensor en este caso es el nodo mqtt in con el topic "TEMPERATURA/LM35" y así dar una respuesta adecuada al led que se está comunicando con mqtt out con el topic "NIVEL/TEMPERATURA". Los valores del PID fueron obtenidos en la herramienta del Matlab PIDTOOL, fueron ensayados en el WEPC observando así su comportamiento.

Figura 4. 10 PID Tuner Matlab para Planta de Temperatura



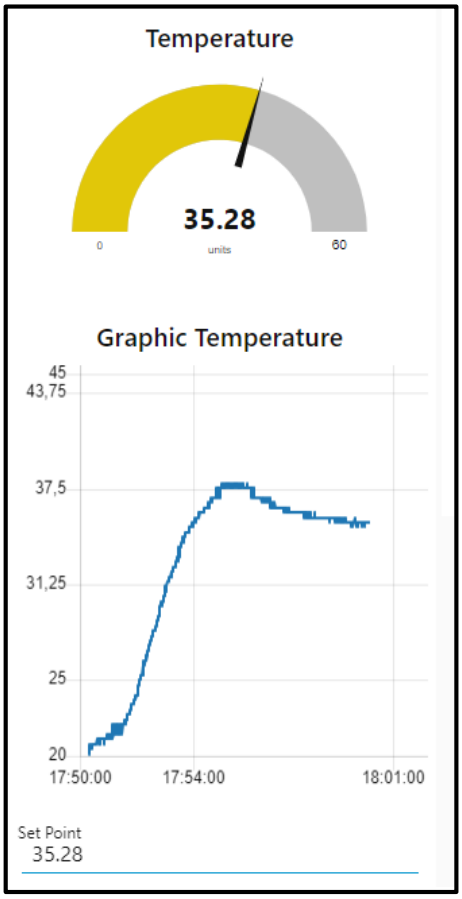
Elaborado por: Pablo Sunta y Diana Yánez

Figura 4. 11 Valores Kp, Ki, Kd Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

Figura 4. 12 Gráfica y Resultados PID de Temperatura



Elaborado por: Pablo Sunta y Diana Yánez

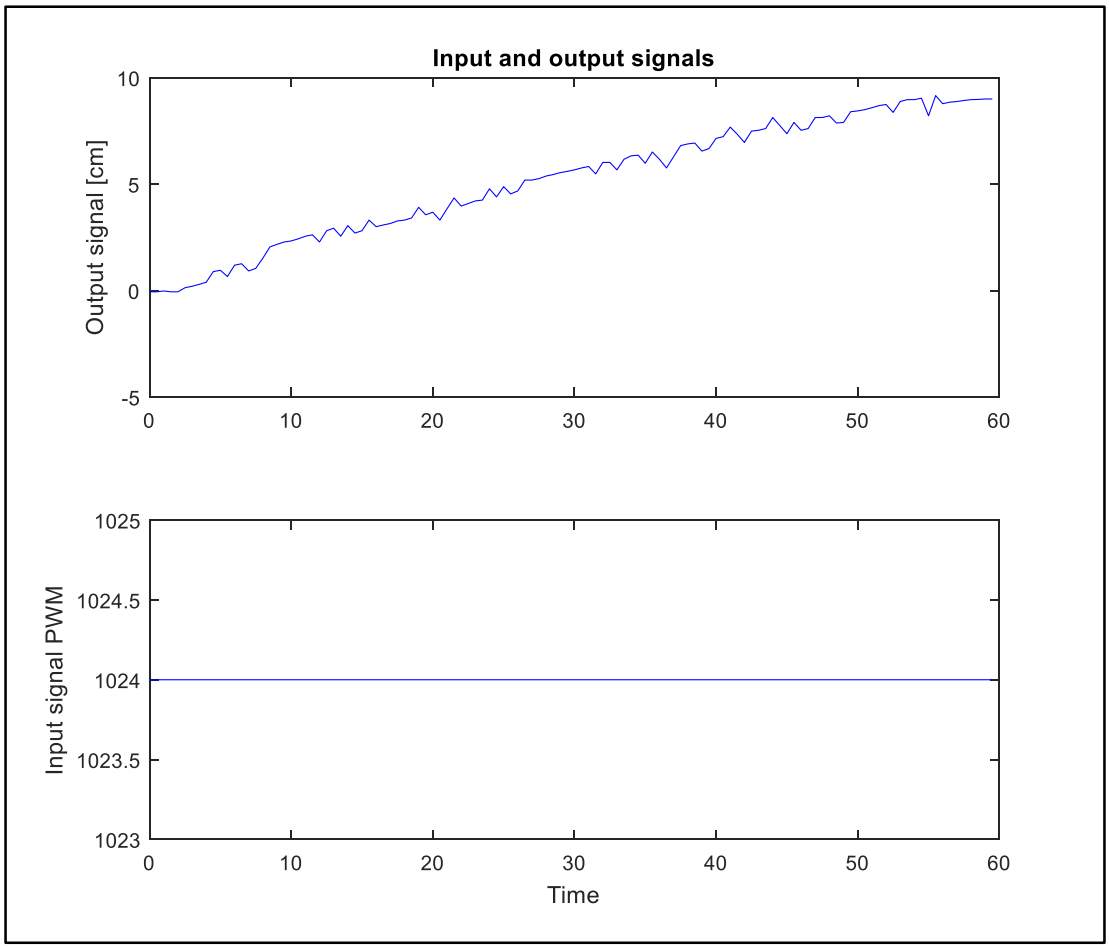
4.3 FUNCIÓN DE TRANSFERENCIA PLANTA NIVEL

Los sistemas de llenado de agua o algún fluido son indispensables en la industria y estos son controlados a través de sensores, actuadores y un controlador automático. El WEPC tiene una planta de nivel de líquidos el cual consta de dos bombas de 5[V], las mismas que son para agregar o retirar líquidos del tanque principal, en la parte superior del mismo está colocado un sensor ultrasónico el cual mide el nivel en centímetros [cm].

Para constituir la estructura y los parámetros que describen matemáticamente a la planta de nivel, se aplica un método de adquisición de datos con IDE de Arduino y usando el WEPC en su forma externa, donde se coloca un set point de 9 [cm] el cual se obtiene gracias al sensor HC-SR04, los datos son recopilados por Arduino el cual procesa las señales eléctricas, les transforma a centímetros [cm] y los imprime por el puerto serial los datos cada 500 [mS].

La función de transferencia se obtiene con una señal de entrada PWM y así accionando la bomba de llenado del tanque principal y recopilando los datos desde el sensor ultrasónico. Para obtener la señal de salida que es el nivel de líquido del tanque principal en centímetros, se toman los datos, se los almacena en una hoja de cálculo y se los importan al IDENT de Matlab para obtener así las gráficas y su respectiva función de transferencia.

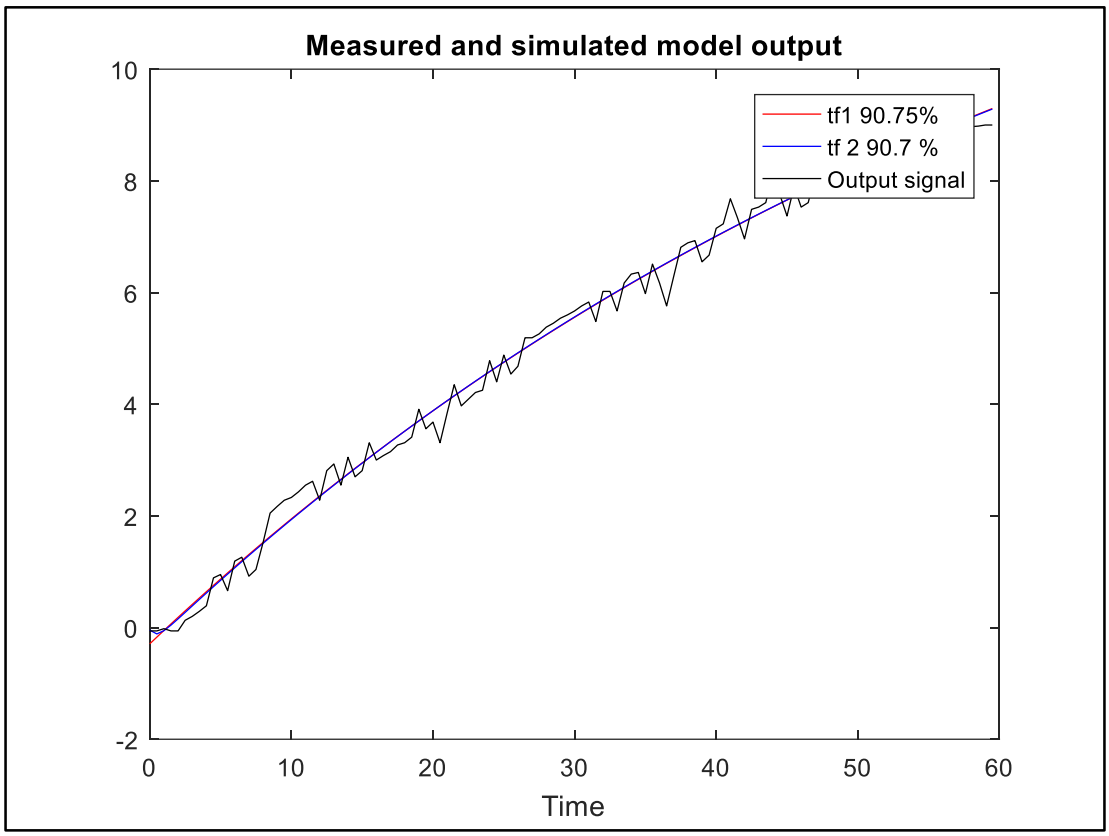
Figura 4. 13 Datos y Set Point Planta de Nivel



Elaborado por: Pablo Sunta y Diana Yánez

De la planta de nivel se obtuvo dos funciones de transferencia (tf1) con dos polos, y un cero, (tf2) con un polo y un cero, como se observa en la figura 4.15 la función de transferencia tf1 tiene una mayor aproximación con un 90.75%.

Figura 4. 14 Función de transferencia Planta de Nivel



Elaborado por: Pablo Sunta y Diana Yánez

$$tf1 = \frac{-0.0004902\ s + 0.0005227}{s^2 + 2.26\ s + 0.03332} \qquad Ec(5)$$

$$tf1 = \frac{-0.0002844\ s + 0.0002305}{s + 0.01447} \qquad Ec(6)$$

CAPÍTULO 5
ANTECEDENTES

5.1 CONCLUSIONES

El entrenador de planta de control inalámbrico WEPC contiene cinco plantas de control basados en procesos industriales y convencionales los cuales esta distribuidos en 3 controladores NodeMCU ESP8266, el primero con las plantas de velocidad, posición angular; el segundo con temperatura y nivel; y finalmente el tercero ejemplifica el área de domótica, cada una de las plantas cuenta con todos los elementos y condiciones electrónicas para que pueda trabajar de forma inalámbrica y externa. De manera que, el usuario tenga un ahorro de tiempo, dinero y no asuma la necesidad de empezar de cero en cuanto a hardware se refiere así se concentre en el aprendizaje e investigación de los diferentes sistemas de control que se pueden implementar en el WEPC.

El protocolo utilizado en el presente proyecto para la comunicación inalámbrica entre el NodeMCU ESP8266 y Raspberry Pi 3 es Message Queing Telemetry Transport (MQTT) destacando su sencillez, ligereza en cuanto bajo consumo de energía y ancho de banda mínimo y tiene facilidad para trabajar con el patrón publicador y subscriptor, es por eso que lo hace adecuado para las aplicaciones con IoT, las cuales actualmente son muy utilizadas y con mucho potencial para su futuro desarrollo.

La Raspberry Pi 3 es un mini ordenador de fácil acceso, bajo coste y consumo el cual se usó como servidor o bróker para gestionar procesar y administrar los datos de los sistemas de control, se optó por el software Iot denominado Node-RED el cual es un instrumento muy potente que sirve para comunicar hardware y servicios de una forma muy rápida y sencilla, además cuenta con una interfaz gráfica y un amplio repertorio de librerías.

Finalmente se trabajó en un proceso real en las plantas de velocidad y temperatura, obteniendo sus respectivas funciones de transferencias y analizándolas para así aplicar un correcto control PID, presentando resultados que cumplen con las características propias de cada variable y propias del controlador, dando así más seguridad y fiabilidad a la hora de ejercer control inalámbrico sobre cualquiera de los procesos que comprende el entrenador.

5.2 RECOMENDACIONES

El WEPC tiene un amplio rango para trabajo, y se puede aprender varios métodos de control industrial en el entrenador, se recomienda a los usuarios elaborar prácticas cada vez más complejas, y realizarlas con diferentes métodos, con el fin de tener más opciones al momento de practicar la ingeniería.

En cuanto a Hardware, se invita desarrollar e implementar más plantas de control y así poder aprovechar todos los pines y capacidades del NodeMCU ESP8266, a su vez diseñar una protección para los elementos del WEPC y poder analizar de mejor manera las variables sin factores externos que interfieren en el control.

Con respecto a las comunicaciones se recomienda hacer también un estudio, investigación más a fondo de la latencia, velocidad y seguridad que existe en el envío y recepción de datos, desde el entrenador al broker para seguir incrementando su eficiencia, y aplicar estos conceptos en cualquier tipo de control automatizado

Detrás de Node-Red hay una gran comunidad dando soporte, actualizaciones, y desarrollo de más aplicaciones se podría incluir una investigación basado en Node-Red para enlazarlo con varias herramientas de internet como son Alexa, IFTTT, Google entre otro tipo de APIs que nos dan apertura para múltiples proyectos de automatización con IOT.

Considerando que el WEPC contiene los procesos principales de control, con este dispositivo se espera que docentes, estudiantes y más usuarios sigan motivando el estudio, investigación e implementación de la Automatización y Control con el internet de las cosas, para seguir desarrollando esa área en el país.

REFERENCIAS

Archila Cordova, D. M. (2013). Estado del arte de las redes de sensores inalambricos. *TIA*, 4.

Barrio, M. (2018). *Internet de las cosas.* Madrid: REUS.

Carballar Falcón, J. A. (2010). *WI-FI Lo que se necesita conocer.* Madrid: Grupo Ramirez.

Contreras, L., Tristancho, J., & Vargas, L. (2015). *Automatizacion en nivel de control de planta mediante el uso de herramientas libres y computacion.* Caldas: Redes de Ingenieria.

Del Valle Hernández, L. (20 de Junio de 2017). *Programar Facil*. Obtenido de NodeMCU tutorial paso a paso desde cero: https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

Del Valle Hernández, L. (4 de Abril de 2019). *Programar Facil*. Obtenido de Novedades de Node-RED 0.20: https://programarfacil.com/blog/raspberry-pi/novedades-node-red-0-20/

Gaviño, I. R. (2010). *Introducción a los Sistemas de Control.* Mexico: Pearson Education.

Gil, I. G. (2018). *Comparativa teórica y práctica de Midlewares y MQTT.* Bilbao.

Gonzales, C. P. (2016). Deteccion y segumiento de objetos por colores en una plataforma raspberry. *Industriales*, 49-50.

Hillar , G. C. (2017). *MQTT Essentials - A Lightweight IoT Protocol.* Birmingham: Packt Publishing.

Laborda, J. (1 de Octubre de 2016). *GitHub.* Obtenido de Introduccion al ESP8266 y NodeMCU: https://github.com/jaimelaborda/Planta-Twittera/wiki/1.-Introducci%C3%B3n-al-ESP8266-y-NodeMCU

Llamas , L. (17 de Abril de 2019). *LUIS LLAMAS*. Obtenido de ¿Qué es MQTT? Su importancia como protocolo IoT: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

Morales Menéndez, R., & Ramírez Mendoza, R. (2013). *Sistemas de control moderno. Volumen I: Sistemas de tiempo continuo.* Monterrey: Digital, Tecnologico de Monterrey.

National Instruments. (4 de 2016). *Entrenador de planta de control.* Obtenido de Datalights: https://www.datalights.com.ec/site2/images/EPC/epc%20manual%20de%20usu

ario%20v3160404.pdf

Pillajo, C., & Hincapie , R. (2018). *Wireless Network Control Sistems de la teoria a la práctica.* Quito, Ecuador: Universitaria Abya-Yala.

Rallo, D. (7 de Marzo de 2019). *Outsourcing Automation Systems*. Obtenido de OASYS: https://oasys-sw.com/3-nuevas-tendencias-en-los-sistemas-de-automatizacion/

Torrente, O. (2016). *ARDUINO. Curso practico de formación.* Madrid: Grupo RC.

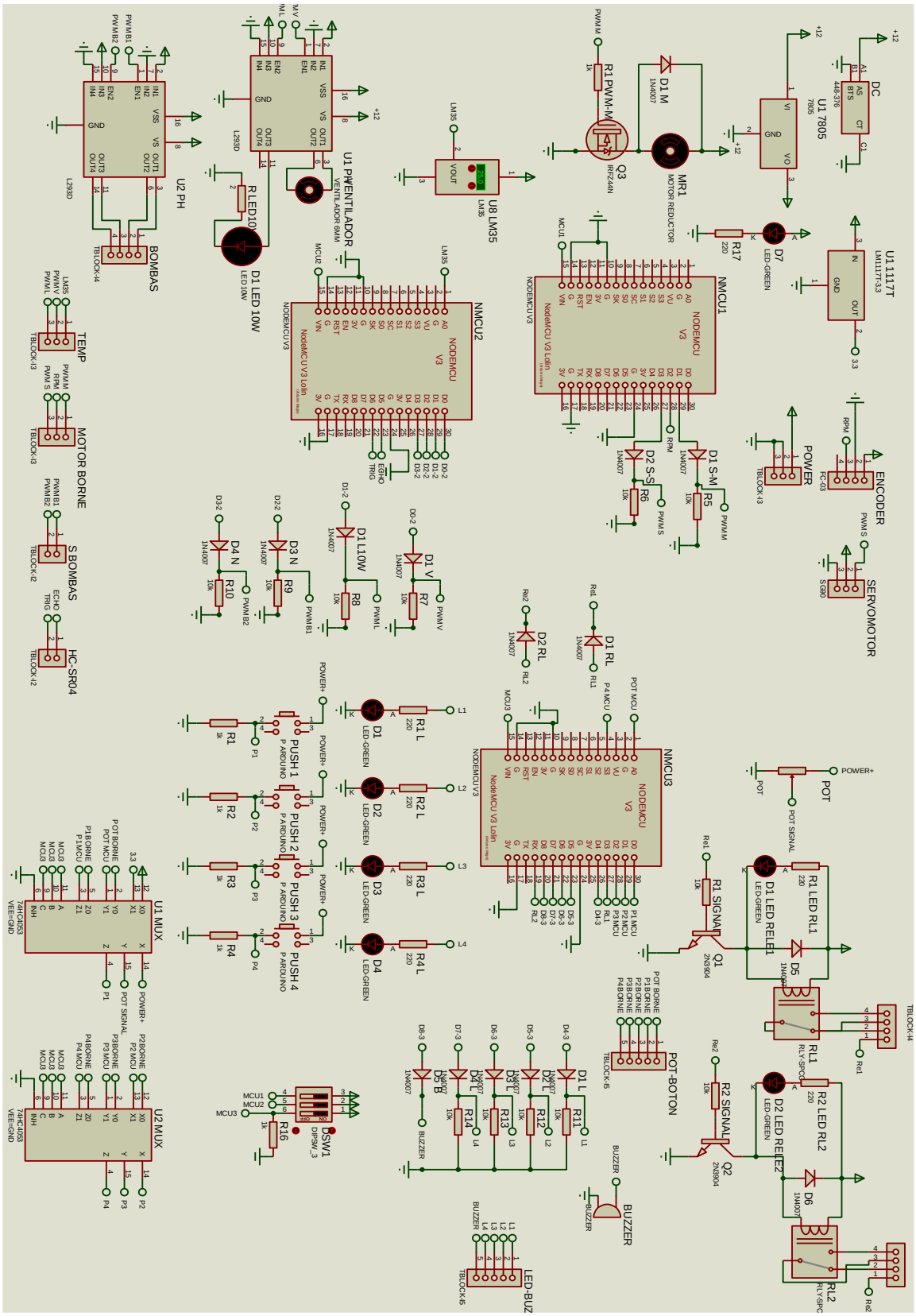
Vázquez, J., Cardona, J., & Leal, J. (2015). *Automatización Neumatica.* Bogota: Ediciones de la U.

Yuan, M. (4 de Agosto de 2017). *IBM*. Obtenido de Conozca MQTT: https://developer.ibm.com/es/articles/iot-mqtt-why-good-for-iot/

ANEXOS

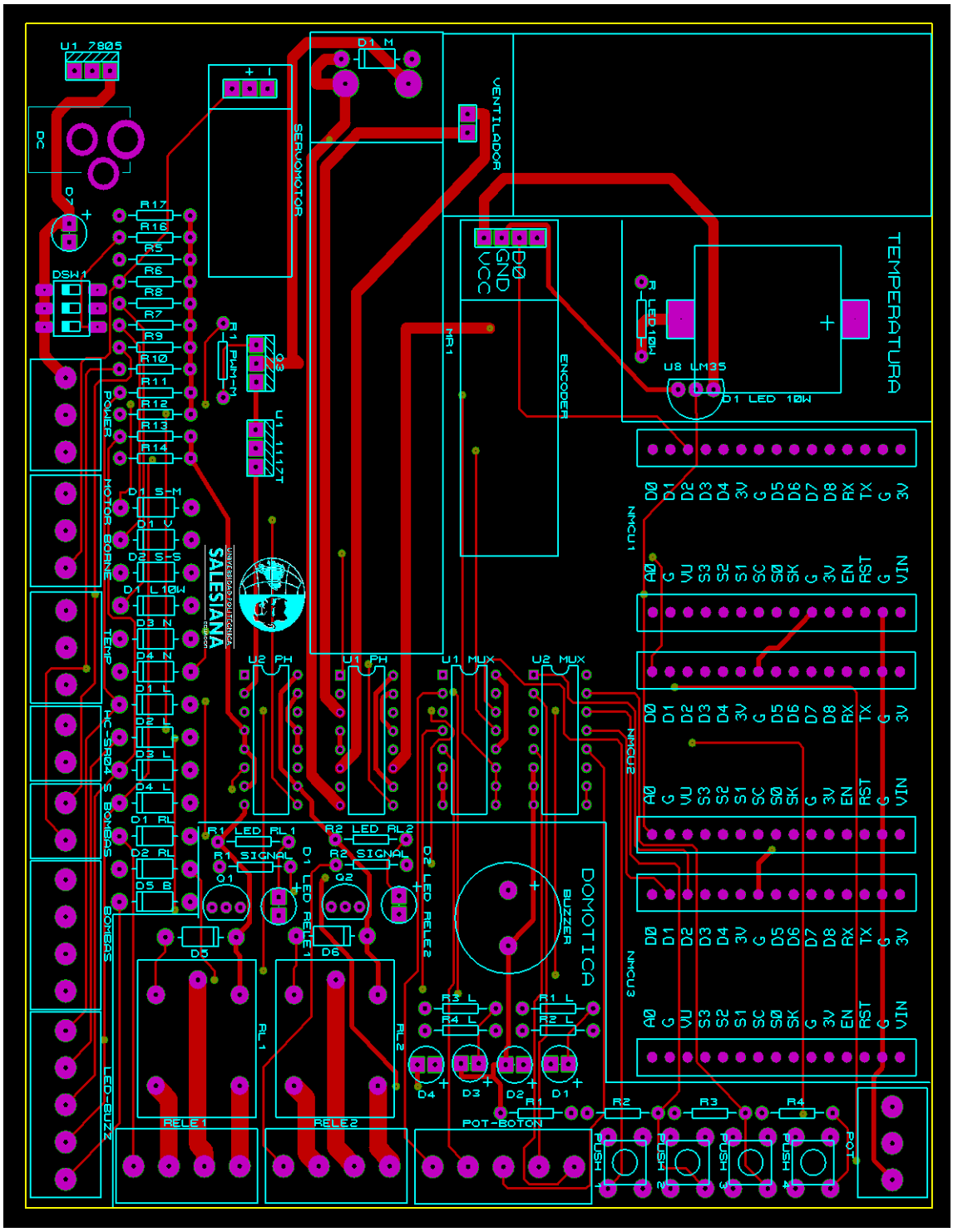
ANEXO 1

Circuito del WEPC



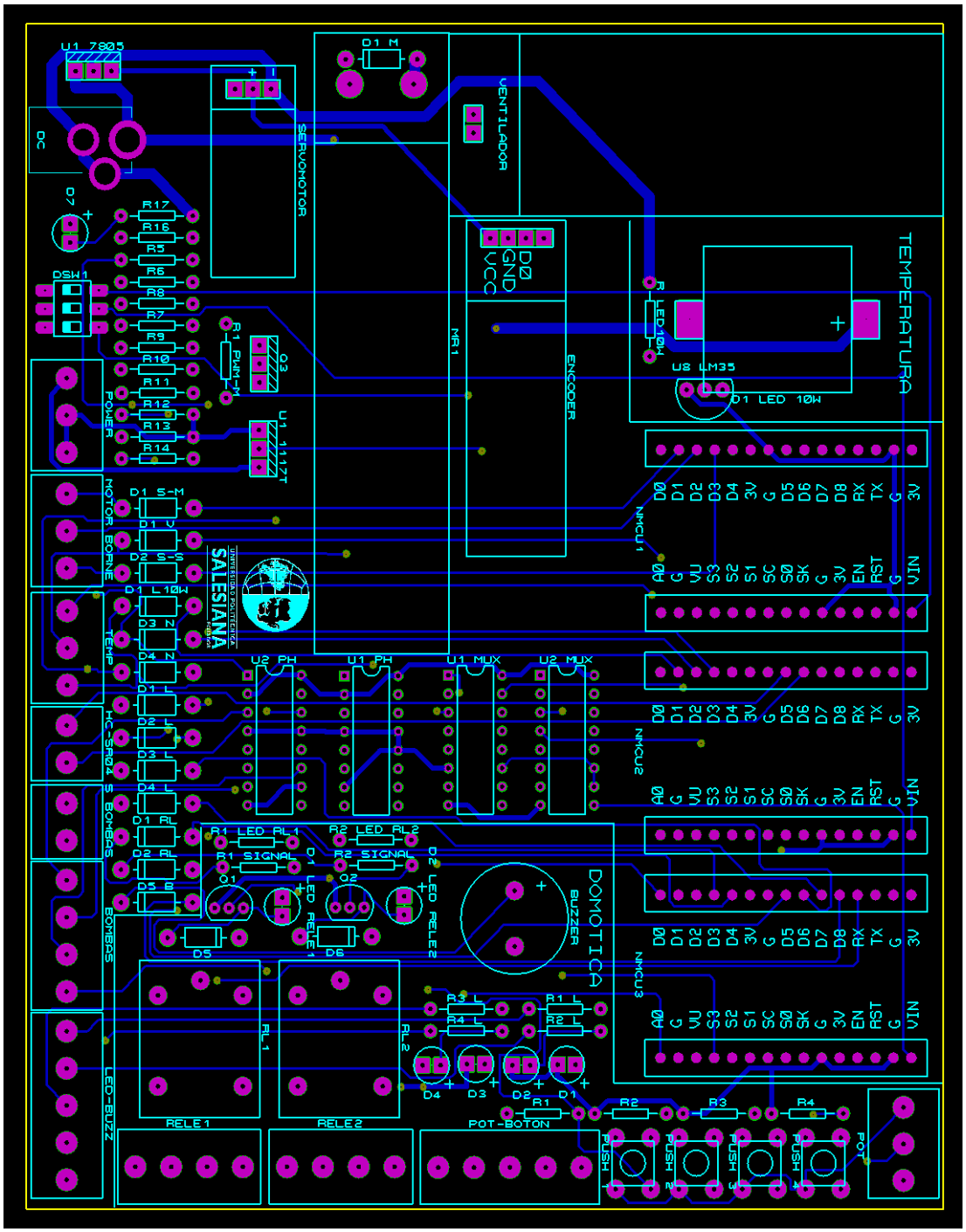
ANEXO 2

PCB TOP COPPER



ANEXO 3

PCB BUTTOM COPPER



ANEXO 4

Entrenador inalámbrico de plantas de control "WEPC"

Manual de operaciones

Capítulo I:

Introducción El Entrenador inalámbrico de Planta de Control "WEPC" es una placa electrónica que incluye varios sensores y actuadores típicos en los sistemas de instrumentación y control tales como temperatura, velocidad, posición, nivel, domótica, señales analógicas de corriente continua, alterna y tren de pulsos.

Este manual explica la forma básica de utilización del WEPC, pero no pretende ser un texto de teoría de control, instrumentación, o programación de Node-Red.

El WEPC está diseñado para conectar a un mini ordenador mediante una red inalámbrica WIFI. El WEPC incluye varios programas escritos en la plataforma de programación Node-Red para analizar y controlar los experimentos.

El objetivo de este equipo es facilitar el aprendizaje de conceptos de teoría de control e instrumentación e internet de las cosas al poner a disposición del usuario varios experimentos prácticos listos para usar. De esta forma se minimiza el tiempo de diseño y construcción electrónico, se asegura la compatibilidad de los sensores con los experimentos, y se obtiene una experiencia de primera mano con las características y problemas de los sistemas físicos reales tales como ruido, precisión, acoplamiento AC/DC, etc. en lugar de usar simulaciones por computadora. Además, habilita la metodología de Aprendizaje Activo (aprender por medio del desarrollo de proyectos prácticos) que aporta significativamente al aprendizaje que usando exclusivamente medios teóricos tales como libros de texto, dictados, y resolución de ejercicios.

Finalmente, el WEPC es una herramienta diseñada para maximizar el aprendizaje de Internet de las cosas y adquisición de datos al proporcionar plantas físicas reales que

funcionan con señales típicas.

Los experimentos que contiene el WEPC son los siguientes:

- Control inalámbrico manual de Velocidad de Motor DC y posición angular
- Control inalámbrico manual de Temperatura y nivel
- Control inalámbrico manual de Domótica

Qué Se Necesita Para Empezar

Para desarrollar las prácticas se necesita:

- Requerimientos de Hardware
  - Entrenador inalámbrico de Planta de Control WEPC
  - Fuente de poder AC/DC de 12V, 1200Ma
  - Mini ordenador Raspberry Pi
  - Destornillador pequeño
  - Router Wifi
  - Computador con cualquier tipo de sistema operativo
- Requerimientos de Software
  - Sistema operativo en Raspberry "Raspberry Pi OS with desktop and recommended software"
  - Conexión a internet estable
  - Servidor Mosquitto en Raspberry Pi
  - Node-red actualizado en Raspberry Pi
  - Un conjunto de nodos de tablero para Node-RED: node-red-dashboard
  - Controlador de lazo PID para Node-RED: node-red-contrib-pid 1.1.6
- Prerrequisitos
  - Conocimientos básicos de programación en JavaScript
  - Conocimientos básicos de sistemas de adquisición de datos y sensores
  - Para las prácticas de control, conocimientos básicos de teoría de control moderno
  - Conocimientos básicos de Redes Inalámbricas

Diagrama Esquemático Y Conexiones Básicas

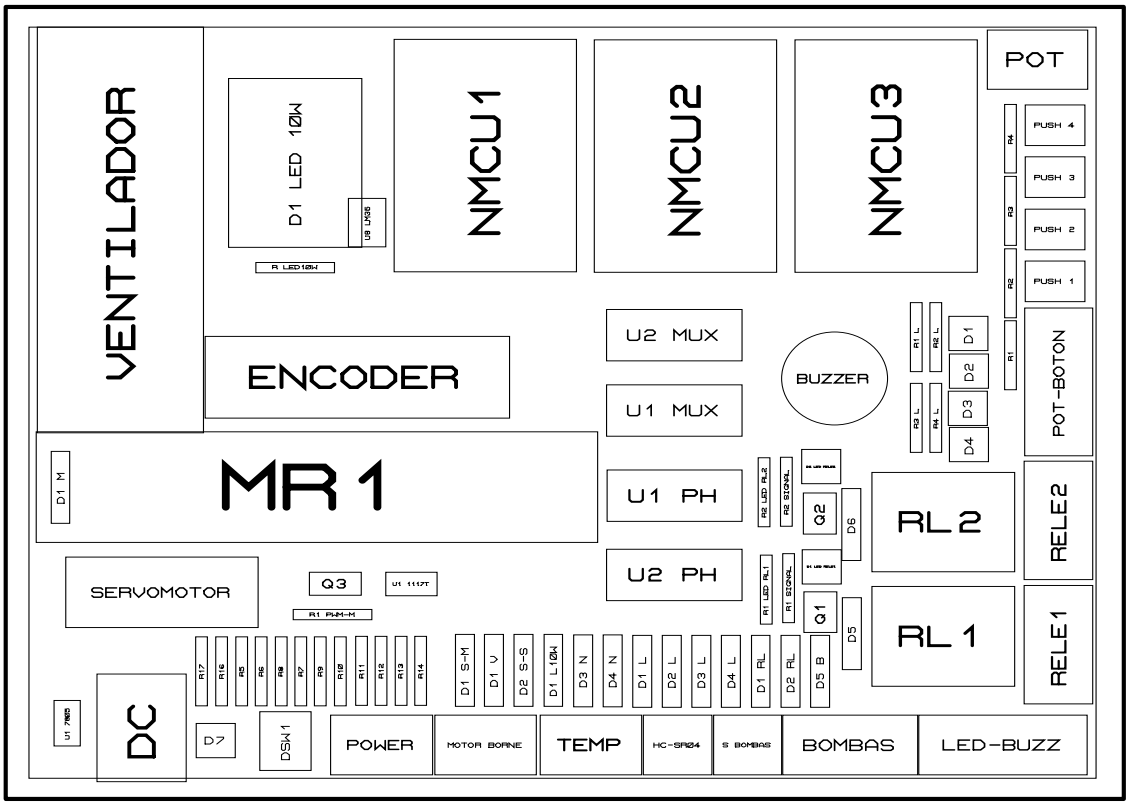
El WEPC y cada una de sus tarjetas tiene dos modos de operación el cual se elige en el Dip Switch DSW1 correspondiente ON: Inalámbrico Off: Externo

Inalámbrico: Usando los módulos Node MCU ESP 8266 y controlar todo desde la Raspberry Pi vía Wifi mediante el protocolo MQTT

Externo: Que es conectando en las borneras del WEPC un controlador externo por ejemplo Arduino, Microcontrolador, cualquier tipo de tarjeta o sistema electrónico diseñado por el usuario y así poder usar todos los actuadores y sensores del WEPC sin la necesidad de desmontar el NodeMCU ESP 8266

NOTA IMPORTANTE: Antes de conectar una fuente de poder al equipo lea completamente esta sección del manual, pues dependiendo del equipo de control que esté usando, puede encenderse en condiciones que conduzcan a calentamiento y posibles daños a los circuitos, e incluso leves quemaduras en la piel.

A continuación, se presenta un diagrama esquemático ubicando las partes principales del equipo.



En el modo externo todos los actuadores y sensores digitales funcionan con lógica directa es decir se activan en 5v y se desactivan en 0v, los sensores analógicos varían entre 5v y 0v

Los Relés tiene la configuración NO (por sus siglas en inglés, Normally Open), cada uno tiene un led indicador D5 y D6.

Comunicaciones WEPC:

La Raspberry Pi es el Broker el que recibe los datos de los NodeMCU los procesa, toma decisiones y después las envía a los NodeMCU. Por lo tanto, necesita tener una IP fija-

Cada uno de los NodeMCU es un cliente, deben tener diferentes nombres para el correcto envió de datos y no colapse la red. Tener muy claro cuál es el topic de cada una de las variables

Capitulo II: Medición y control manual de Velocidad y Posición Angular

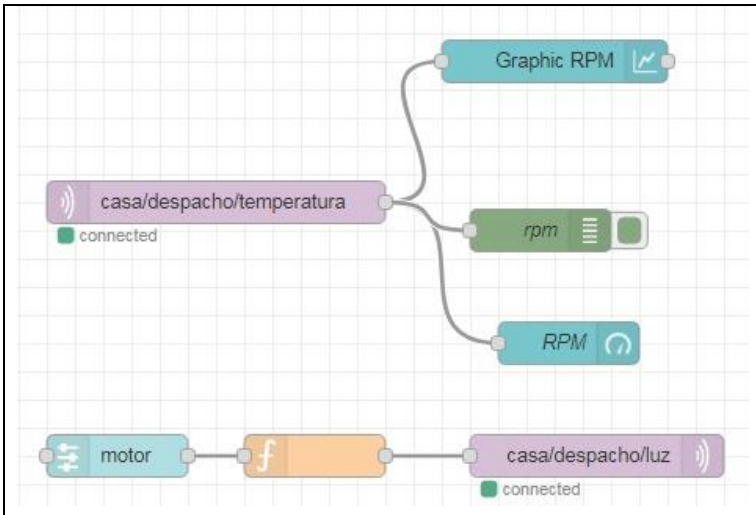
El EPC incluye un motor de corriente continua (Motor DC) en cuyo eje está acoplado un encoder de 20 pulsos por revolución para medir la velocidad. El motor es controlado por una señal de voltaje DC que puede variar entre 0 y 5 voltios en lógica directa. La salida del encoder es una señal pulsante.

Control manual de Velocidad y Posición angular

Colocar una IP fija a la Raspberrry Pi: 192.168.0.107

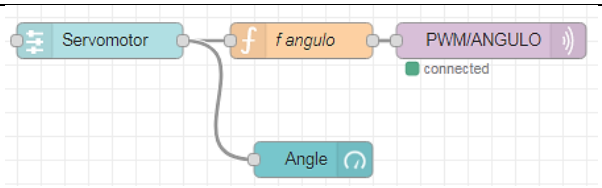
Planta Motor-Servomotor		
Client: Motor/Servo		
Topic (in)	MOTOR/RPM	La velocidad del Motor en RPM
Topic (out)	PWM/ANGULO	El valor de PWM del motor entre 0-1024 acompañado del carácter “A” ejemplo: “A1024” El valor del ángulo del servomotor entre 0-180 acompañado del carácter “B” ejemplo: “B1024”

Programación Node-Red

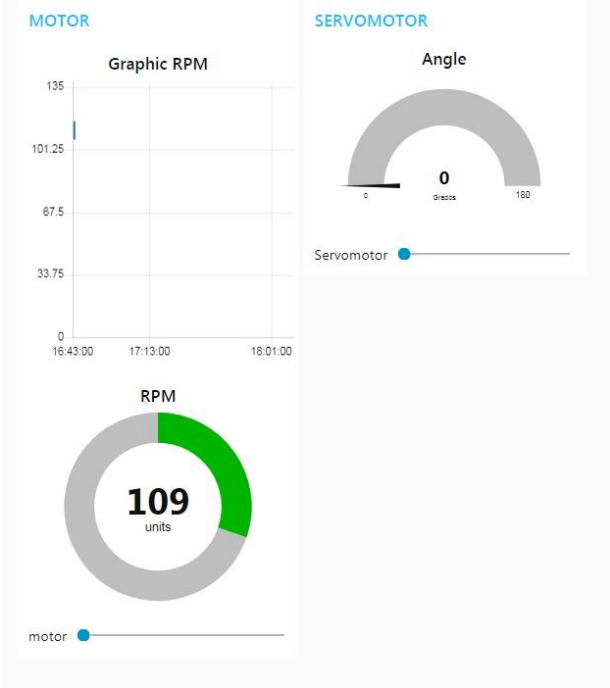


Código de la función:

```
var x=msg.payload;
x='A'+x;
msg.payload=x;
return msg;
```

	Código de la función “f angulo”:
	var x=msg.payload;
	x='B'+x;
	msg.payload=x;
	return msg;

Dashboard Node-Red



- Con el Slide “Motor” se controla manualmente la velocidad del Motor DC
- Con el Slide “Servomotor” se controla manualmente el ángulo del servo

En el NodeMCU se carga el siguiente código:

#include <ESP8266WiFi.h> #include <PubSubClient.h>

```
#include <Servo.h>

Servo myservo;
// Update these with values suitable for your network.
const char* ssid = "ine4c";
const char* password = "ine4c4000";
const char* mqtt_server = "192.168.0.107";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
int x=0;
int servo=0;
int pwm=0;
int s=0;
String n;
String oldm;
String m;
//------------------------------ Variables de motor Izquierdo ----------------------------
int N = 20;                    // nùmero de ranuras del encoder
float diametro = 6.8;          // diametro de la llanta cm
int contadorTicks = 3;         // nùmero de ticks para calculo de velocidad
int tam = 10;                  // tamaño del vector del calculo de promedio, se debe descomentar la linea que se vaya a usar

static volatile unsigned long debounce = 0;
volatile unsigned muestreoActualInterrupcionL = 0;  // variables para definiciòn del tiempo de interrupciòn y calculo de la velocidad motor Izquierdo
volatile unsigned muestreoAnteriorInterrupcionL = 0;
double deltaMuestreoInterrupcionL = 0;
uint8_t encoderL = D2;     // pin de conexiòn del encoder Izquierdo
int llantaL = D1;     // pin de conexiòn de llanta Izquierda   (pin de PWM)
double frecuenciaL = 0;
// frecuencia de interrupciòn llanta Izquierda
double Wl = 0;                 // Velocidad angular L
double rpm=0;
double r=0;
int CL = 0;                    // contador Ticks
float vectorL[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};     // vector de almacenamiento de datos para promedio del tiempo de interrupciones
float Y=0;
double S=Y;
float alpha =0.12;
//------------------------------ Variables de motor Izquierdo ----------------------------
```

```
--------------------
void ICACHE_RAM_ATTR counter ();
int R=0;
void setup() {
attachInterrupt(digitalPinToInterrupt(en
coderL),counter,RISING);          //
linea para añadir una interrupciòn a un
PIN
  Serial.begin(115200);
  ////////////////////////////////////////////////
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
 ////////////////////////////////////////////////
  myservo.attach(0,544,2400);
}
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi
network
  Serial.println();
  Serial.print("Connecting to ");
Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() !=
WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void callback(char* topic, byte*
payload, unsigned int length) {

  if((char)payload[0] == 'B'){
     m="";
    }
  if((char)payload[0] == 'A'){
     n="";
    }
//  Serial.print("Message arrived [");
//  Serial.print(topic);
//  Serial.print(topic);
//  Serial.print("] ");

  for (int i = 1; i < length; i++) {
   //Serial.println((char)payload[i]);
   if((char)payload[0] == 'A'){
     n=n+((char)payload[i]);
    }
    if((char)payload[0] == 'B'){
     m=m+((char)payload[i]);
    }
  }
  pwm=n.toInt();
  servo=m.toInt();
  //Serial.println(n);
  //Serial.println(m);
//  Serial.println(pwm);
  //pwm=0.8046*pwm+200;
  analogWrite(D1,pwm);
  myservo.write(servo);
  oldm=m;
//  if ((char)payload[0] == '1') {
```

```
//   digitalWrite(BUILTIN_LED, LOW);
// Turn the LED on (Note that LOW is
the voltage level
//    // but actually the LED is on; this is
because
//    // it is acive low on the ESP-01)
//  } else {
//              digitalWrite(BUILTIN_LED,
HIGH);  // Turn the LED off by making
the voltage HIGH
//  }
}
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT
connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client"))
{
      Serial.println("connected");
      // Once connected, publish an
announcement...
client.publish("casa/despacho/temperatu
ra", "Enviando el primer mensaje");
      // ... and resubscribe
client.subscribe("casa/despacho/luz");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5
seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void counter() {        // funciòn de
interrupciòn del enconder llanta derecha
  if(      digitalRead (encoderL) &&
(micros()-debounce > 1000)) {
    CL++;
    if (CL == contadorTicks){
      float media = 0;
      deltaMuestreoInterrupcionL =
muestreoActualInterrupcionL -
muestreoAnteriorInterrupcionL;        //
diferencia tiempos de interruciones de
ticks del motor
      for(int i=0;i < tam-1;i++){
// relleno del vector para calculo
posterior del promedio
        vectorL[i]=vectorL[i+1];
      }
      vectorL[tam-
1]=deltaMuestreoInterrupcionL;
// ùltimo dato del vector
      for(int i=0;i<tam;i++){
// Càlculo de la media del vector
        media = vectorL[i]+ media;
      }
      media = media/tam;
      deltaMuestreoInterrupcionL =
media;                                //
se reemplaza por el valor de su medìa.
      frecuenciaL = (1000)/
deltaMuestreoInterrupcionL;
// frecuencia de interrupciòn
```

```
// velocidad lineal cm/s
        muestreoAnteriorInterrupcionL = muestreoActualInterrupcionL;
// se actualiza el tiempo de interrupciòn anterior
        CL = 0;
      }
      debounce = micros();
 }
}
void loop() {
   if(pwm<50){
     N = 20;                          // nùmero de ranuras del encoder
     diametro = 6.8;                  // diametro de la llanta cm
     contadorTicks = 2;               // nùmero de ticks para calculo de velocidad
     tam = 10;                        // tamaño del vector del calculo de promedio, se debe descomentar la linea que se vaya a usar
     debounce = 0;
     muestreoActualInterrupcionL = 0; // variables para definiciòn del tiempo de interrupciòn y calculo de la velocidad motor Izquierdo
     muestreoAnteriorInterrupcionL = 0;
     deltaMuestreoInterrupcionL = 0;
     encoderL = D2;   // pin de conexiòn del encoder Izquierdo
     llantaL = D1;       // pin de conexiòn de llanta Izquierda   (pin de PWM)

     frecuenciaL = 0;                 // frecuencia de interrupciòn llanta Izquierda
     Wl = 0;                          // Velocidad angular L
     rpm=0;
     r=0;
     CL = 0;                          // contador Ticks
    //  vectorL[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};      // vector de almacenamiento de datos para promedio del tiempo de interrupciones
     Y=0;
     S=Y;
     alpha =0.12;
   }
   muestreoActualInterrupcionL = millis();          // se asigna el tiempo de ejecuciòn a el muestreo actual
                                    // velocidad lineal cm/s
     Wl = contadorTicks*((2*3.141516)/N)*frecuenciaL;          // frecuencia angular Rad/s
     rpm=(Wl*60)/(2*3.1415);
     if(pwm<50)
     rpm=0;
     Y=rpm;// velocidad lineal cm/s
     S=(alpha*Y)+((1-alpha)*S);
           // PWM de la llanta derecha
```

```
   analogWrite(llantaL,pwm);
// PWM de la llanta izquierda
                  // se muestra el tiempo
entre TIC y TIC
    Serial.print(rpm);
    Serial.print(",");
     Serial.print(r);// se muestra el tiempo
entre TIC y TIC
   Serial.print(",");
//  long now = millis();
//  if (now - lastMsg > 40) {
    Serial.println(S);// se muestra el
tiempo entre TIC y TIC
//    }
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
 long now = millis();
  if (now - lastMsg > 50) {
    lastMsg = now;
    R=int(S);
    snprintf (msg, 75, "%1.1f", S);
//    //Serial.print("Publish message: ");
//    //Serial.println(msg);

client.publish("casa/despacho/temperatu
ra", msg);
    Serial.println(S);
  }
```

Capitulo III: Medición y control manual de Temperatura y Nivel

El EPC incluye en su interior un LED de Temperatura de alto brillo que produce calor cuando se enciende. Este elemento simula un dispositivo de calentamiento tipo On/Off como puede ser una niquelina, el led y el ventilador está conectado en lógica directa controlado por una señal de 0v o 5v

El sensor de temperatura LM35 convierte la señal de calor en una señal de voltaje según la siguiente ecuación. ℃ = V ∗ 1024/3300

Dónde: ℃ es la temperatura en grados Celsius

- *V* es el voltaje que entrega el sensor de temperatura
- 100 es una constante numérica

Un Ventilador instalado frente al LED de Temperatura permite ingresar aire al EPC, introduciendo también una perturbación en el sistema térmico y provocando

enfriamiento forzado.

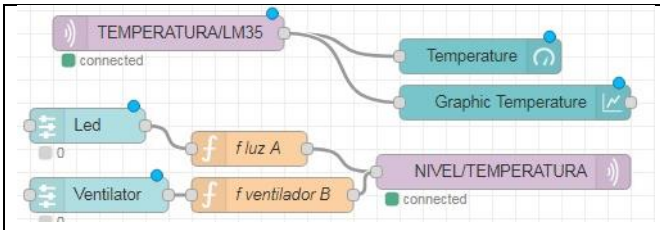
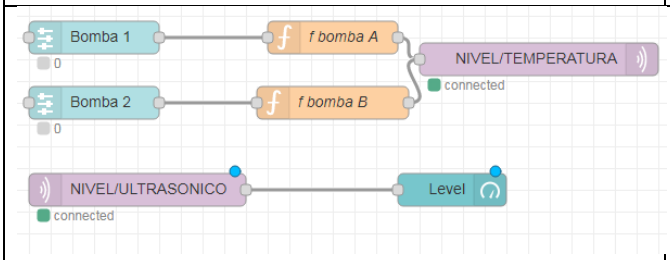
La planta de nivel cuenta con las borneras para conectar dos bombas de 5v una de llenado y otra de vaciado, de igual manera para conectar un sensor ultrasónico HC-SR04, la fórmula para detectar la altura del líquido depende del tanque principal utilizado.

Control manual de Velocidad y Nivel

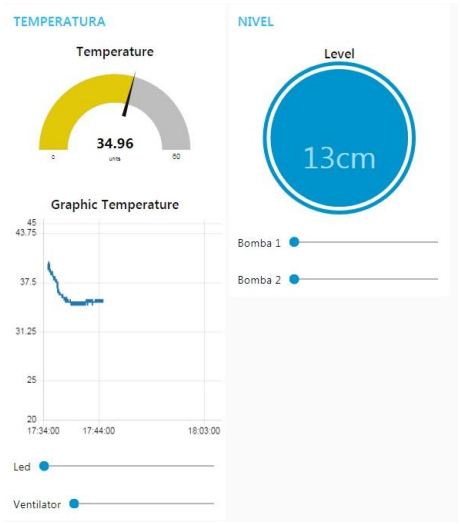
Colocar una IP fija a la Raspberrry Pi: 192.168.0.107

Planta: Temperatura - Nivel		
Client: NIVEL/TEMPERATURA		
Topic (in)	TEMPERATURA/LM35	La temperatura del LM35 del WEPC en grados centígrados
Topic (in)	NIVEL/ULTRASONICO	El nivel del tanque de agua medido por el sensor ultrasónico HC SR04 en centímetros
Topic (out)	NIVEL/TEMPERATURA	El valor de PWM del led para aplicar calor en el WEPC entre 0-1024 acompañado del carácter “A” ejemplo: “A1024” El valor de PWM del ventilador para enfriar el WEPC entre 0-1024 acompañado del carácter “B” ejemplo: “B1024” El valor de PWM de la primera bomba de agua para ingresar liquido al tanque principal del WEPC entre 0-1024 acompañado del carácter “C” ejemplo: “C1024” El valor de PWM de la segunda bomba de agua encargada de sacar liquido del tanque principal del WEPC entre 0-1024 acompañado del carácter “D” ejemplo: “D1024”

Programación Node-Red

<table>
<tr>
<td> 
</td>
<td>Código de la función “f luz A”:
var x=msg.payload;
x='A'+x;
msg.payload=x;
return msg;
Código de la función “f ventilador B”:
var x=msg.payload;
x='A'+x;
msg.payload=x;
return msg;</td>
</tr>
<tr>
<td> 
</td>
<td>Código de la función “f bomba A”:
var x=msg.payload;
x='C'+x;
msg.payload=x;
return msg;
Código de la función “f bomba B”:
var x=msg.payload;
x='D'+x;
msg.payload=x;
return msg;</td>
</tr>
</table>

Dashboard Node-Red



- Con el Slide “Led” se controla manualmente la intensidad del Led
- Con el Slide “Ventilador” se controla manualmente la velocidad del ventilador
- Con el Slide “Bomba 1” se controla manualmente la velocidad de la Bomba 1
- Con el Slide “Bomba 2” se controla manualmente la velocidad de la Bomba 2

En el NodeMCU ESP8266 se carga el siguiente código:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.
const char* ssid = "ine4c";
const char* password = "ine4c4000";
const char* mqtt_server = "192.168.0.107";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

long lastMsg1 = 0;

//////////////////////
int luz=0;
int ventilador=0;
int b1=0;
int b2=0;
```

```
String n;
String oldm;
String m;
String o;
String p;
//////////////////
const int trigPin = 14;  //D4
const int echoPin = 12;  //D3
long duration;
float distance;
float d=0;
//////////////

void setup() {
  pinMode(BUILTIN_LED,  OUTPUT);
// Initialize the BUILTIN_LED pin as an
output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);


  analogWriteFreq(500);
  pinMode(trigPin,  OUTPUT);    // Sets
the trigPin as an Output
  pinMode(echoPin, INPUT);   // Sets the
echoPin as an Input
}

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi
network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while          (WiFi.status()          !=
WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte*
payload, unsigned int length) {

  if((char)payload[0] == 'B'){
     m="";}
  if((char)payload[0] == 'A'){
     n="";}
  if((char)payload[0] == 'C'){
     o="";}
  if((char)payload[0] == 'D'){
     p="";}


  for (int i = 1; i < length; i++) {
   if((char)payload[0] == 'A'){
```

```
    n=n+((char)payload[i]);}
   if((char)payload[0] == 'B'){
    m=m+((char)payload[i]);}
   if((char)payload[0] == 'C'){
    o=o+((char)payload[i]);}
   if((char)payload[0] == 'D'){
    p=p+((char)payload[i]);}

  }
  luz=n.toInt();
  ventilador=m.toInt();
  b1=o.toInt();
  b2=p.toInt();

//  Serial.println(luz);
//  Serial.println(ventilador);
//  Serial.println(b1);
//  Serial.println(b2);

  analogWrite(D0,ventilador);
  analogWrite(D1,luz);
  analogWrite(D2,b1);
  analogWrite(D3,b2);


}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
   Serial.print("Attempting MQTT
connection...");
   // Attempt to connect
   if (client.connect("Client-
Temperatura/Nivel")) {
    Serial.println("connected");
    // Once connected, publish an
announcement...

client.publish("TEMPERATURA/LM3
5", "Enviando el primer mensaje");
    // ... and resubscribe

client.subscribe("NIVEL/TEMPERAT
URA");
   } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5
seconds");
    // Wait 5 seconds before retrying
    delay(5000);
   }
  }
}

void loop() {

  analogWrite(D0,ventilador);
  analogWrite(D1,luz);
  analogWrite(D2,b1);
  analogWrite(D3,b2);
  if (!client.connected()) {
   reconnect();
  }
  client.loop();

  long now = millis();
```

```
  //----------------- Temperatura----------
-----
  if (now - lastMsg > 300) {
  //Serial.print("Distance: ");
  //Serial.println(distance);
    int analogValue = analogRead(A0);
    float millivolts =
(analogValue/1024.0) * 3300; //3300 es
el voltaje con que se alimenta el sensor
    int celsius = millivolts/10;
    float celsius1= millivolts/10-5;
    lastMsg = now;
    snprintf (msg, 75, "%1.2f", celsius1);
    Serial.println(celsius1);

client.publish("TEMPERATURA/LM3
5", msg);

  }
//------------NIVEL--------------------
    long now1 = millis();
   if (now1 - lastMsg1 > 500) {
    digitalWrite(trigPin, LOW);   // Clears
the trigPin
    delayMicroseconds(2);
    digitalWrite(trigPin,  HIGH);  // Sets
the trigPin on HIGH state for 10 micro
seconds
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Reads the echoPin, returns the sound
wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    distance=   (duration/29.14)/2;        //
Calculating the distance
    d=13.00-distance;
    //Serial.print("Distance:                ");
Serial.println(d);
      lastMsg1 = now1;
      snprintf (msg, 75, "%1.1f", d);

client.publish("NIVEL/ULTRASONIC
O", msg);
    }
}
```

Capitulo IV: Medición y control Domótica

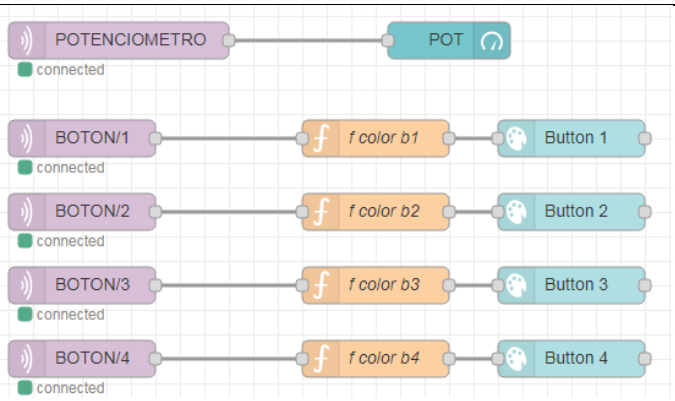
La Planta de domótica está conformado por cinco sensores, un potenciómetro y cuatro botones. Además, tiene siete actuadores cuatro luces led, dos relés de uso común, un buzzer. Los mismos que pueden ser controlados inalámbrica por el Node-Red o externamente por las borneras.

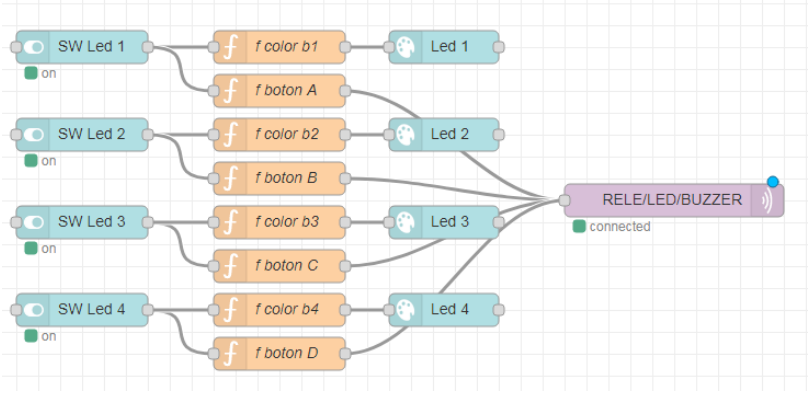
Control manual de Domótica

Colocar una IP fija a la Raspberrry Pi: 192.168.0.107

Planta: Domótica		
Client: Domotica		
Topic (in)	BOTON/1	El estado on/off del botón 1 del WEPC
Topic (in)	BOTON/2	El estado on/off del botón 2 del WEPC
Topic (in)	BOTON/3	El estado on/off del botón 3 del WEPC
Topic (in)	BOTON/4	El estado on/off del botón 4 del WEPC
Topic (in)	POTENCIOMETRO	El valor analógico del potenciómetro limitado entre valores de 0-1024
Topic (out)	RELE/LED/BUZZER	El estado del led 1 ON=1, OFF=0 acompañado del carácter “A” ejemplo: “A1” El estado del led 2 ON=1, OFF=0 acompañado del carácter “B” ejemplo: “B1” El estado del led 3 ON=1, OFF=0 acompañado del carácter “C” ejemplo: “C1” El estado del led 4 ON=1, OFF=0 acompañado del carácter “D” ejemplo: “D1” El estado del Relé 1 ON=1, OFF=0 acompañado del carácter “E” ejemplo: “E1” El estado del Relé 2 ON=1, OFF=0 acompañado del carácter “D” ejemplo: “D1” El estado del Buzzer ON=1, OFF=0 acompañado del carácter “F” ejemplo: “F1”

Programación Node-Red





Función "f botón A":

var x=msg.payload;

x='A'+x;

msg.payload=x;

return msg;

Función "f botón B":

var x=msg.payload;

x='B'+x;

msg.payload=x;

return msg;

Función "f botón C":

var x=msg.payload;

x='C'+x;

msg.payload=x;

return msg;

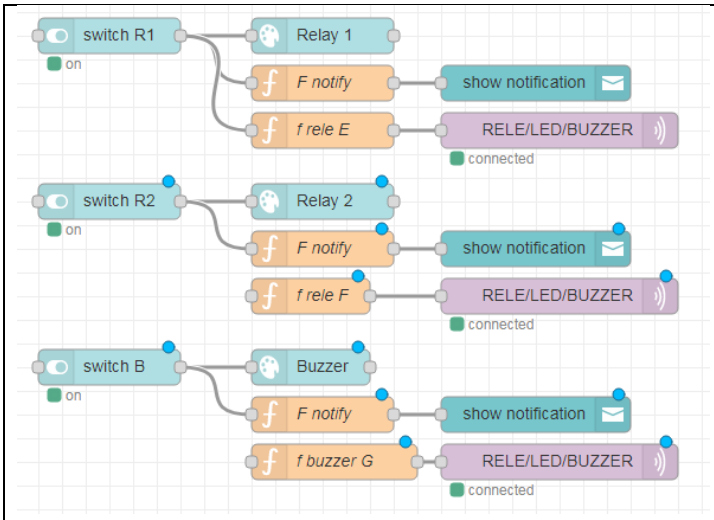
Función "f botón D":

var x=msg.payload;

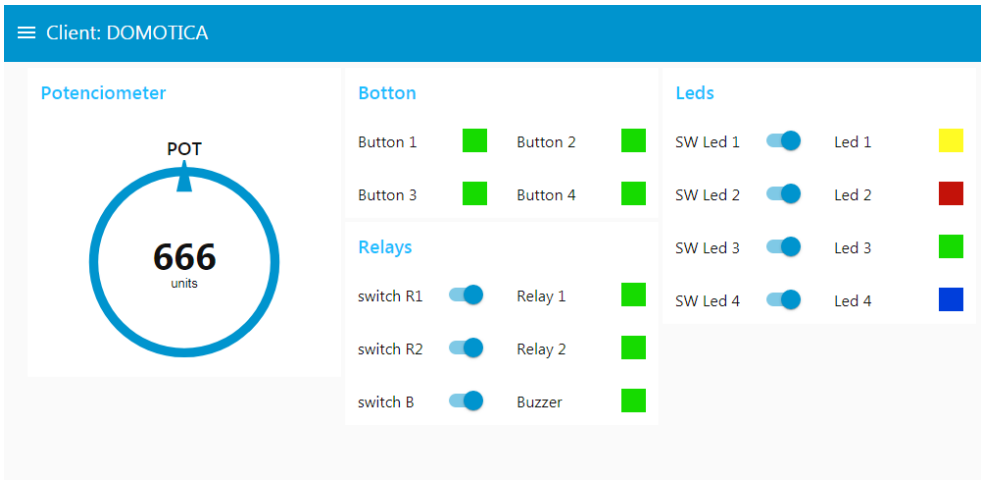
x='D'+x;

msg.payload=x;

return msg;

<table>
<tr>
<td> 
</td>
<td>Función “f rele E”:
var x=msg.payload;
x='E'+x;
msg.payload=x;
return msg;
Función “f rele F”:
var x=msg.payload;
x='F'+x;
msg.payload=x;
return msg;
Función “f buzzer G”:
var x=msg.payload;
x='G'+x;
msg.payload=x;
return msg;</td>
</tr>
</table>

Dashboard Node-Red



En el NodeMCU cargar el siguiente código:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "ine4c";
const char* password = "ine4c4000";
const char* mqtt_server = "192.168.0.107";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
int dato=0;

//////////////////////
String n;
String m;
String o;
String p;
int boton1 = 0;
int boton2 = 0;
int boton3 = 0;
int boton4 = 0;
const int led1 = D4;
const int led2 = D5;
const int led3 = D6;
const int led4 = D7;

const int buzzer = D8;
const int rele1 = 3;
const int rele2 = D3;
//////////////

void setup() {
  pinMode(BUILTIN_LED,  OUTPUT);
// Initialize the BUILTIN_LED pin as an
output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  ///////////////////
  pinMode(D0, INPUT);
  pinMode(D1, INPUT);
  pinMode(D2, INPUT);
  pinMode(10, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(rele1, OUTPUT);
  pinMode(rele2, OUTPUT);
  }

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi
network
  Serial.println();
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {

  if((char)payload[0] == '1'){
     m="";}
  if((char)payload[0] == '2'){
     n="";}
  if((char)payload[0] == '3'){
     o="";}
  if((char)payload[0] == '4'){
     p="";}
  for (int i = 1; i < length; i++) {
    if((char)payload[0] == '1'){
      n=n+((char)payload[i]);}
    if((char)payload[0] == '2'){
      m=m+((char)payload[i]);}
    if((char)payload[0] == '3'){
      o=o+((char)payload[i]);}
    if((char)payload[0] == '4'){
      p=p+((char)payload[i]);}
  }
}
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("Client-Domotica")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("CASA/BOTON", "Enviando el primer mensaje");
      // ... and resubscribe
      client.subscribe("RELE/LED");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
  boton1 = digitalRead(D0);
  boton2 = digitalRead(D1);
  boton3 = digitalRead(D2);
  boton4 = digitalRead(10);
  if (!client.connected()) {
```

```
  reconnect();
 }
 client.loop();
 if(boton1 == HIGH)
digitalWrite(led1, HIGH);
 else if(boton1 == LOW)
digitalWrite(led1, LOW);
 if(boton2 == HIGH)
digitalWrite(led2, HIGH);
 else if(boton2 == LOW)
digitalWrite(led2, LOW);

 if(boton3 == HIGH)
digitalWrite(led3, HIGH);
 else if(boton3 == LOW)
digitalWrite(led3, LOW);

 if(boton4 == HIGH)
digitalWrite(led4, HIGH);
 else if(boton4 == LOW)
digitalWrite(led4, LOW);
```

```
  value = analogRead(A0);
 Serial.println(value);
 if(value<340){
  digitalWrite(buzzer, HIGH);
  digitalWrite(rele1, HIGH);
  digitalWrite(rele2, HIGH);
 }
 if(value>340 && value<682){
  digitalWrite(buzzer, LOW);
  digitalWrite(rele1, LOW);
  digitalWrite(rele2, HIGH);
 }
 if(value>682){
  digitalWrite(buzzer, LOW);
  digitalWrite(rele1, HIGH);
  digitalWrite(rele2, LOW);
 }
 delay(100);
}
```

Especificaciones Generales

Señales de Medición y control

Entradas Digitales	**Salidas de Tren de Pulsos**
Motor DC 1	Encoder 1
Servomotor 1	Sensor Ultrasónico HC-SR04 1
LED de Temperatura. 1	Botón 4
Ventilador 1	
Bomba 2	**Total: 6**

Relé 2 Led 4 Buzzer 1 Total: 13 0-5VDC	0-5 VDC
Salidas Analógicas: Temperatura 1 Potenciómetro 1 **Total: 2** Nivel 0-5 VDC Consumo 5mA	

ALIMENTACIÓN

12 VDC, 2000 mA para alimentación del WEPC, motores y componentes electrónicos

DIMENSIONES FÍSICAS

Largo 16cm x Ancho 13cm